Tampereen yliopisto

Viivi Siuko

KÄYTTÄJÄKESKEISYYS KETTERÄSSÄ OHJELMISTOKEHITYKSESSÄ

Tekniikan ja luonnontieteiden tiedekunta
Diplomityö
Marraskuu 2019

TIIVISTELMÄ

Viivi Siuko: Käyttäjäkeskeisyys ketterässä ohjelmistokehityksessä
Diplomityö
Tampereen yliopisto
Tietojohtamisen diplomi-insinöörin tutkinto-ohjelma
Tarkastajat: Tenure track -tutkija Henri Pirkkalainen ja professori Marko Seppänen
Marraskuu 2019

Ketterässä ohjelmistokehityksessä keskitytään yhteistyöhön asiakkaan kanssa ja ohjelmiston toimivuuteen, käyttäjien osallistamisen ja käytettävyyteen panostamisen sijaan. Käytettävyysongelmat huomataankin usein liian myöhään, käyttöönoton jälkeen, jolloin ongelmien korjaaminen on kallista ja käyttäjät ovat voineet jo pettyä ohjelmistoon. Ohjelmistoon pettyminen voi johtaa siihen, että käyttäjät eivät halua käyttää ohjelmistoa enempää kuin on pakollista. Tähän ongelmaan on aiemmassa kirjallisuudessa kehitetty ratkaisuksi viitekehys, jonka avulla käyttäjäkeskeinen ohjelmistosuunnittelu ja ketterä ohjelmistokehitys voidaan yhteensovittaa. Näiden kahden menetelmän yhteensovittaminen tarkoittaa ketterämpää ohjelmistosuunnittelua ja käyttäjäkeskeisempää ohjelmistokehitystä. Aiemmin ehdotetut viitekehykset eivät kuitenkaan toimi kaikissa organisaatioissa ja projekteissa: Viitekehysten toteuttamiseen tarvitaan kaksi tiimiä, joista toinen keskittyy ohjelmistokehitykseen ja toinen ohjelmistosuunnitteluun. Lisäksi viitekehykset on luotu käyttäjäkeskeisen ohjelmistosuunnittelun näkökulmasta, eli kuinka sitä voidaan toteuttaa paremmin ketterissä projekteissa. Kaikissa organisaatioisa ja projekteissa ei ole resursseja kahden tiimin hyödyntämiseen, eikä myöskään aiempaa osaamista käyttäjäkeskeisestä ohjelmistosuunnittelussa.

Tässä tutkimuksessa luodaan uusi käyttäjäkeskeisen ohjelmistosuunnittelun ja ketterän ohjelmistokehityksen viitekehys, jonka lähtökohtana on käyttäjäkeskeisyyden lisääminen ketterässä ohjelmistokehityksessä. Viitekehyksen tarkoitus on sopia mahdollisimman erilaisiin projekteihin, joten viitekehys kootaan siten, että viitekehystä noudattavat projektit on mahdollista toteuttaa yhdellä tiimillä. Aiemmassa kirjallisuudessa on tunnistettu käyttäjäkeskeisen ohjelmistosuunnittelun ja ketterän ohjelmistokehityksen yhteensovittamisen periaatteet, jotka toimivat viitekehyksen pohjana. Ketteristä ohjelmistokehitysmenetelmistä valittiin scrum ohjelmistokehityksen iteraation malliksi, sillä scrum on yritysten keskuudessa suosituin ketterä ohjelmistokehitysmenetelmä. Uusi viitekehys eroaa scrumin viitekehyksestä siten, että ennen iteraatioita, tai sprinttejä, järjestetään erillinen suunnitteluvaihe, iteraatio 0, ja seuraavien iteraatioiden aikana järjestetään käytettävyyspalavereita ja asiakkaan lisäksi myös käyttäjät kutsutaan katselmointeihin. Uuden viitekehyksen toimivuutta arvioidaan eräässä ohjelmistoalan yrityksessä, haastattelemalla yrityksen henkilöstöä eri projekteista. Haastatteluissa selvitetään viitekehyksen hyötyjä ja haasteita, sekä kerätään kehitysehdotuksia.

Lopullinen viitekehys on aiempaa joustavampi, sillä siinä kerrotaan vaihtoehtoisia tapoja lisätä käyttäjäkeskeisyyttä projekteihin ja sen toteuttamiseen ei tarvita kahta tiimiä. Pienissä projekteissa voi riittää, että käyttäjät kutsutaan vain katselmointeihin, vaikka se ei poista riskiä toiminnallisuuksien uudelleen toteuttamisesta, sillä käytettävyysongelmat huomataan vasta toteutuksen jälkeen. Etenkin suuremmissa projekteissa on kannattavaa panostaa käytettävyysongelmien korjaamiseen jo ennen toteutuksen aloittamista, mikä voidaan tehdä osallistamalla käyttäjiä käytettävyyspalavereihin ja iteraatioon 0.

Avainsanat: Käyttäjäkeskeinen ohjelmistosuunnittelu, ketterä ohjelmistokehitys

ABSTRACT

Viivi Siuko: User-centred design in agile software development
Master of Science Thesis
Tampere University
Master's Degree Programme in Information and Knowledge Management
Examiners: Tenure Track Researcher Henri Pirkkalainen and Professor Marko Seppänen
November 2019

The focus of agile software development is in cooperation with customer and developing working software, instead of involving customers and making software usable. Problems in usability are often noticed only after the implementation of the new software, when fixing the problems is expensive and the users may have already gotten disappointed with the software. Getting disappointed with the software, may lead users to avoid using the software any more than they need to. A framework for integrating user-centred systems design and agile software development has been suggested for this problem in previous research. Merging these two methods results in more agile systems design and more user-centred software development. However, the frameworks proposed before, are not suitable for all organizations and projects as there needs to be two teams, one of which concentrates on design and the other on software development. In addition, the frameworks have been created from design perspective, concentrating on how user-centred systems design can be better put in to practice in agile projects. There are not enough resources for engaging two teams in one project nor previous experience in user-centred systems design in every organization.

In this research, a new framework is created for integrating user-centred systems design and agile software development. The framework is created from the perspective of agile software development as the aim is to find out how it can be done in a more user-centred way. The new framework aims to be suitable for different kinds of projects and to be practicable with only one team. There are principles for integrating user-centred systems design and agile software development, gathered in previous literature. These principles and scrum, a method of agile software development, form the base of the new framework. Before every iteration, or sprint, there is a planning phase called iteration 0 in the new framework. In the following iterations, usability planning meetings are added, and instead of inviting only customers to iteration reviews, also the users are invited. The new framework is evaluated by interviewing employees from different projects of a software company. The objective with the interviews is to discover the benefits and challenges of the framework, and to gather ideas for developing the framework even further.

The final framework is more flexible than the previous ones as it contains different options for increasing the level of user-centricity in agile projects, and there is no need for two teams when the framework is put to practice. In small projects, it may be enough to add user-centricity by inviting users to iteration reviews, even though it does not decrease the risk of having to fix usability problems after development. Especially in bigger projects it is beneficial to invest in correcting the usability problems in planning phase by engaging users in usability planning meetings and iteration 0.

Keywords: User-centred systems design, agile software development

ALKUSANAT

Tämä tutkimus on toteutettu osana kohdeyrityksen trainee-jaksoa, ja tutkimuksen aihe on noussut kohdeyrityksen halusta palvella asiakkaitaan paremmin. Käyttäjäkeskeinen ketterä ohjelmistokehitys mahdollistaa käyttäjien huomioimisen aiempaa paremmin projektien aikana, jolloin asiakkaille voidaan toimittaa paremmin käytettäviä ohjelmistoja. Tämän tutkimuksen tavoitteena on ollut selvittää, kuinka käyttäjäkeskeistä ketterää ohjelmistokehitystä voidaan toteuttaa kohdeyrityksen kaltaisissa yrityksissä.

Tutkimuksen toteutuksen aikana sain ohjausta sekä yliopiston että kohdeyrityksen puolesta. Haluankin kiittää kaikkia ohjaajiani todella hyvästä ohjauksesta ja heidän antamastaan tuesta. Lisäksi haluan kiittää Tietojohtajakilta Man@geria ja killan jäsenistöä opiskelijaelämän hauskuuttamisesta ja kaikesta, mitä olen hallitusvuosien aikana oppinut. Opintojen ja kurssiassistenttina vietetyn vuoden aikana opituista asioista haluan kiittää Tietojohtamisen yksikköä.

Perhettäni ja ystäviäni kiitän kaikesta tuesta ja kannustamisesta, jota olen saanut opintojen aikana. Erityisesti haluan kiittää rakasta avopuolisoani Mattia tuesta, kannustamisesta, kärsivällisyydestä ja innoittamisesta. Kiitos myös Novalle, joka muistutti koko diplomityöprosessin ajan taukojen ja ulkoilun tärkeydestä.

Tampereella, 20.11.2019

Viivi Siuko

SISÄLLYSLUETTELO

KUVALUETTELO

TAULUKKOLUETTELO

LYHENTEET JA MERKINNÄT

Asiakas	Ohjelmistoa ostavan yrityksen edustaja tai edustajat, jotka ovat tiiviisti mukana projektin toteutuksessa
Ketterä	Ketterä ohjelmistokehitys, muutoksiin mukautuvien ja iteratiivisten ohjelmistokehitysmenetelmien kattokäsite, ks. scrum
Käyttäjä	Ohjelmiston loppukäyttäjä, joka ei ole tiiviisti mukana projektin toteutuksessa
Käyttäjäkeskeinen	Käyttäjäkeskeinen ohjelmistosuunnittelu keskittyy käytettävyyteen ja korostaa käyttäjien osallistamisen tärkeyttä
Käytettävyys	Ohjelmiston käytettävyys määräytyy sen kyvystä mahdollistaa käyttäjälle tavoitteidensa saavuttaminen mahdollisimman vaivattomasti
Iteraatio	Lyhytkestoinen (kahdesta kuuteen viikkoa) toteutusjakso projektin sisällä, jonka lopputuloksena on valmis ohjelmisto tai ohjelmiston osa
Projektitiimi	Toimittajan projektitiimi, joka suunnittelee ja toteuttaa ohjelmiston
Scrum	Yleisimmin yrityksissä käytössä oleva ketterä ohjelmistokehitysmenetelmä

1. JOHDANTO

Ketterässä ohjelmistokehityksessä ohjelmiston toimivuutta ja muutoksiin mukautuvuutta arvostetaan enemmän kuin käytettävyyttä ja etukäteen tapahtuvaa suunnittelua (Agile Alliance 2001; Ferreira et al. 2007a; Lievesley & Yee 2006; Singh 2008). Ketterässä ohjelmistokehityksessä ei siis huomioida käytettävyyttä tarpeeksi (Armitage 2004), eikä käytettävyyden suunnittelulle ole varattu aikaa (Salah et al. 2014). Lisäksi käyttäjien osallistamisen sijaan, ketterässä ohjelmistokehityksessä toiminnallisuuksia toteutetaan asiakkaan toiveiden ja palautteen pohjalta. Asiakkaat eivät kuitenkaan yleensä tiedä käyttäjien todellisia tarpeita, joten käyttäjien osallistaminen ohjelmiston kehityksessä ja ohjelmiston suunnittelu käyttäjistä kerätyn tiedon pohjalta on välttämätöntä ohjelmiston menesetymisen kannalta. (Miller 2005; Fox et al. 2008). Ohjelmiston käytettävyydellä on suuri rooli ohjelmiston menestymisessä, sillä käytettävyysongelmat voivat johtaa siihen, että käyttäjät eivät halua käyttää ohjelmistoa (Fox et al. 2008; Da Silva et al. 2018). Lisäksi käytettävyysongelmat huomataan usein vasta käyttöönoton jälkeen (Kuusinen & Väänänen-Vainio-Mattila 2012), jolloin ongelmien korjaaminen on hyvin kallista (Ferreira et al. 2007b).

Käytettävyysongelmien ratkaisuksi Miller (2005), Sy (2007) ja Fox et al. (2008) ovat ehdottaneet käyttäjäkeskeisen ohjelmistosuunnittelun ja ketterän ohjelmistokehityksen yhteensovittamista, ja luoneet siihen viitekehyksiä. Viitekehysten noudattamiseen tarvitaan kuitenkin kaksi erillistä tiimiä, joista toinen vastaa suunnittelusta ja toinen ohjelmistokehityksestä, mutta kaikissa organisaatioissa ei ole mahdollista käyttää kahta tiimiä yhdessä projektissa pienempien resurssien vuoksi (Miller 2005). Tutkimusten lähtökohtana on ollut, **kuinka käyttäjäkeskeisyyttä voidaan toteuttaa ketterissä projekteissa**, joten tutkimuksissa oletetaan, että organisaatioissa on käytettävyystiimit ja käytettävyysosaamista. Lisäksi viitekehykset ovat keskenään hyvin samankaltaisia ja luotu toistensa pohjalta. Da Silva et al. (2012) ovatkin nostaneet esiin tarpeen uusille viitekehyksille, jotka sopisivat erilaisiin projekteihin ja erilaisille organisaatioille. Koska viitekehyksissä suunnittelu ja ohjelmistokehitys tehdään erillään toisistaan, Da Silva et al. (2018) toivovat, että uudessa viitekehysehdotuksessa käyttäjien tekemät käytettävyyden arvioinnit ja asiakkaiden vastuulla oleva toiminnallisuuksien testaus sovitetaan yhteen.

Ketterässä ohjelmistokehityksessä olevien käytettävyysongelmien ja kirjallisuudesta

nousseiden tarpeiden perusteella tutkitaan, **kuinka ketterää ohjelmistokehitystä voi toteuttaa käyttäjäkeskeisemmin**. Jotta edelliseen tutkimuskysymykseen voidaan vastata, tutkimuksessa selvitetään, **mitkä käytännöt tukevat käyttäjäkeskeisyyden sovittamista ketterään ohjelmistokehitykseen**. Tutkimuksen tavoitteena on luoda viitekehys, joka soveltuu hyvin erilaisiin projekteihin, myös sellaisiin, joissa käytettävyyden suunnitteluun on pienet resurssit ja yrityksiin, joissa käytettävyyteen ei ole aiemmin panostettu. Tutkimuksessä etsitään siis vastauksia myös siihen, **miten käytettävyydestä vastaavan tiimin puuttuminen ja projektien yksilöllisyys vaikuttavat viitekehykseen**.

Tässä tutkimuksessa siis kootaan uusi viitekehys käyttäjäkeskeisen ohjelmistosuunnittelun ja ketterän ohjelmistokehityksen yhteensovittamiselle aiemman kirjallisuuden pohjalta. Viitekehyksen rakentamisessa huomioidaan yhden tiimin asettamat rajoitteet, jonka vuoksi uudesta viitekehyksestä tulee hyvin erilainen kuin aiemmat viitekehykset. Ketterän ohjelmistokehityksen osalta tutkimuksessa keskitytään vain yhteen menetelmään, scrumiin, sillä se on yritysten keskuudessa yleisimmin käytössä oleva ketterä ohjelmistokehitysmenetelmä. Ketterien ohjelmistokehitysmenetelmien käyttöä koskevaan kyselyyn (Version One 2019) vastanneista yrityksistä 72% käytti scrumia tai siihen pohjautuvia menetelmiä.

Tutkimuksessa koottu viitekehys esitellään eräästä yrityksestä valituille ohjelmistokehittäjille, projektipäälliköille ja konsulteille, ja viitekehystä kehitetään heidän haastatteluiden pohjalta. Kaikki haastateltavat henkilöt tekevät töitä projekteissa, joissa valmista ohjelmistoa kustomoidaan asiakkaan tarpeiden mukaan ja jotka toteutetaan yhden tiimin voimin. Haastattelun tarkoituksena on arvioida uuden viitekehyksen toimivuutta käytännössä, mahdollisuutta noudattaa sitä vain yhden tiimin voimin ja soveltuvuutta erilaisiin projekteihin.

Seuraavaksi, luvussa 2, kerrotaan tarkemmin, mitä ketterä ohjelmistokehitys tarkoittaa ja kuinka scrumia noudattavat projektit etenevät. Luvussa 3 esitellään ensin käyttäjäkeskeinen ohjelmistosuunnittelu, kerrotaan, miten ketterä ohjelmistokehitys hyötyy käyttäjäkeskeisyydestä, sekä mitä haasteita yhteensovittamisessa on, ja lopuksi kootaan viitekehys pala palalta aiemman kirjallisuuden pohjalta. Luvussa 4 keskitytään tutkimuksen toteuttamiseen: haastatteluiden toteuttamiseen ja aineiston analysointiin. Luvussa 5 esitellään haastatteluiden tulokset tiivistäen haastateltavien ajatukset eri viitekehyksen osa-alueisiin liittyen. Viimeisessä luvussa vertaillaan kirjallisuudesta ja haastatteluista nousseita aiheita, ja muokataan viitekehystä haastatteluiden pohjalta. Lisäksi viimeisessä lu-

vussa palataan tutkimuskysymyksiin esittäen niihin tutkimuksen aikana saadut vastaukset, kerrotaan tutkimusta rajoittaneista tekijöistä ja ehdotetaan jatkotutkimusaiheita, joita tämän tutkimuksen aikana on noussut esiin.

2. KETTERÄ OHJELMISTOKEHITYS

2.1 Ketterän ohjelmistokehityksen julistus ja arvot

Ketterä ohjelmistokehitys sisältää useita eri menetelmiä, joita yhdistää samat arvot ja periaatteet. Näihin menetelmiin sisältyy muun muassa scrum, XP (*extreme programming*), kanban, lean startup, crystal ja DSDM (*dynamic systems development method*), joista scrumia tai siihen pohjautuvia menetelmiä käyttää suuri osa (72%) VersionOnen kyselyyn vastannaista yrityksistä (VersionOne 2019). Eri menetelmiä edustavat ja perinteisille ohjelmistokehityksen menetelmille vaihtoehtoa etsivät ammattilaiset kokoontuivat yhdessä kokoamaan ketterien menetelmien arvot ketterän ohjelmistokehityksen julistukseen (engl. *The Manifesto for Agile Software Development*) (Highsmith & Cockburn 2001). Siinä todetaan seuraavasti:

”Kokemuksemme perusteella arvostamme:

- **Yksilöitä ja kanssakäymistä** enemmän kuin menetelmiä ja työkaluja
- **Toimivaa ohjelmistoa** enemmän kuin kattavaa dokumentaatiota
- **Asiakasyhteistyötä** enemmän kuin sopimusneuvotteluja
- **Vastaamista muutokseen** enemmän kuin pitäytymistä suunnitelmassa

Jälkimmäisilläkin asioilla on arvoa, mutta arvostamme ensiksi mainittuja enemmän.” (Agile Alliance 2001).

Opelt et al. (2013) mukaan julistus ymmärretään usein väärin etenkin dokumentaation ja sopimusneuvotteluiden osalta: väärinymmärryksistä huolimatta julistuksessa ei tarkoiteta, että dokumentaatiota tai sopimuksia ei tarvitsisi tehdä. Sen sijaan ylimääräinen dokumentointi ja sopimusneuvottelu tulisi karsia pois. Highsmith & Cockburn (2001) tiivistävät, että kun projekteissa pitää joustaa jostain niin on parempi joustaa menetelmien ja työkalujen, dokumentaation, sopimusneuvotteluiden ja suunnitelmien suhteen mieluummin kuin kanssakäymisestä, toimivasta ohjelmistosta, asiakasyhteistyöstä ja muutokseen reagoimisesta, sillä jälkimmäisillä on enemmän arvoa asiakkaalle.

Ketterässä ohjelmistokehityksessä korostetaan ihmisten merkitystä. Yksilöiden kyvykkyyksille ja taidoille annetaan paljon arvoa, sillä ihmisten ollessa tarpeeksi taitavia, he voivat käyttää eritasoisia prosesseja ja saavuttaa tavoitteensa. Sen sijaan hyväkään prosessi ei korvaa ihmisten epäpätevyyttä, sillä prosessit toimivat vain ihmisten toiminnan

tehostajina. Liian tarkkaan määritellyt prosessit voivat myös rajoittaa työskentelyä, sillä prosessit eivät mukaudu muutoksiin samalla tavalla kuin kyvykkäät ihmiset, jos heille on annettu mahdollisuudet käyttää luovuuttaan ja ongelmanratkaisukykyään. (Opelt et al. 2013; Cockburn & Highsmith 2001).

Ketterässä ohjelmistokehityksessä painotetaan myös arvoa, joka syntyy ihmisten välisestä kanssakäymisestä ja siihen liittyvästä luottamuksesta tiimin jäsenten välillä. Luottamuksen kautta tiimin jäsenet jakavat keskenään tärkeää tietoa ja uusia ideoita nopeammin ja kattavammin, ja usein ihmiset yltävät parempiin suorituksiin yhdessä kuin erikseen. Säännöllinen kanssakäyminen vähentää dokumentaation tarvetta ja nopeuttaa muutoksiin reagoimista, etenkin jos kanssakäymiseen luetaan mukaan myös asiakasyhteistyö. Mitä tiiviimmin ja aktiivisemmin asiakkaat ovat projektissa mukana, sitä paremmin lopullinen tuote vastaa asiakkaan tarpeita, myös silloin kun tarpeet muuttuvat projektin aikana. (Opelt et al. 2013; Dingsøyr et al. 2012; Cockburn & Highsmith 2001; Highsmith & Cockburn 2001)

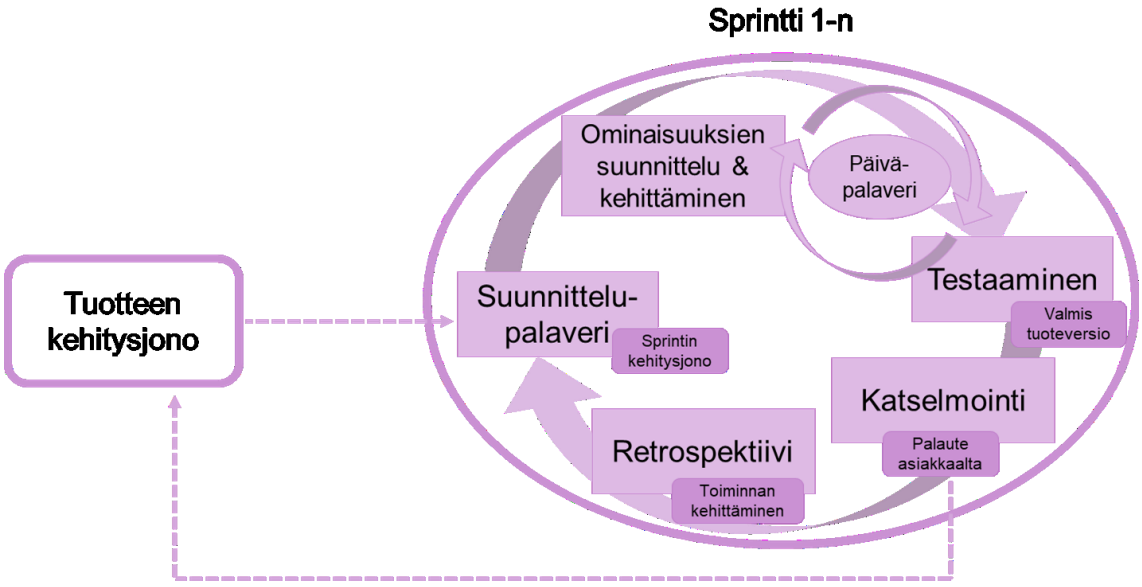
Ketterän ohjelmistokehityksen menetelmissä toistuu käytäntö tehdä ohjelmistokehitystä lyhyissä, noin kahdesta kuuteen viikkoon kestävissä iteraatiojaksoissa. Jokaisen iteraatiojakson päätteeksi tuotetaan valmis ohjelmisto tai ohjelmiston osa, jolloin asiakas, kehittäjät ja muut sidosryhmät näkevät konkreettisesti projektin edistymisen ja voivat antaa toteutetuista ominaisuuksista palautetta. Ennen seuraavan iteraatiojakson alkua päätetään, mitkä ominaisuudet ovat tärkeimpiä toteuttaa seuraavaksi. Ominaisuudet voidaan valita aiemmin suunniteltujen joukosta tai asiakas voi ehdottaa täysin uusia ominaisuuksia. (Highsmith & Cockburn 2001) Ketterän ohjelmistokehityksen ydin tiivistyykin kykyyn toteuttaa muuttuvia toivomuksia ja tarpeita, siis kykyyn reagoida muutokseen nopeasti (Opelt et al 2013; Dingsøyr et al. 2012; Highsmith & Cockburn 2001). Jotta muutokseen voidaan reagoida nopeasti, on tiimin kyettävä nopeisiin päätöksiin, mitä voidaan tukea tarjoamalla tarvittavat asiantuntijat tiimin saataville tai jopa osaksi tiimiä, ja tekemällä päätökset iteraatiovaihekohtaisesti (Cockburn & Highsmith 2001).

Ketterän ohjelmistokehityksen julistus ei kuitenkaan tarjoa yksiselitteistä määritelmää ketteryydelle, joten Conboy (2009) määritteli ketteryyden aiemman kirjallisuuden ja ketterän ohjelmistokehityksen arvojen pohjalta seuraavasti: *Ketteryys on ohjelmistokehitysmenetelmän jatkuva valmius muutoksen luomiseen nopeasti tai vaivattomasti, muutoksen omaksumiseen proaktiivisesti tai reaktiivisesti ja muutoksesta oppimiseen, edistäen samalla koettua asiakasarvoa (taloudellisuus, laatu ja yksinkertaisuus) ja hyödyntäen sisäisiä ja ulkoisia resursseja.*

Seuraavassa alaluvussa käsitellään Scrumia esimerkkinä ketterästä ohjelmistokehityksen menetelmästä, sillä Scrum on yritysten keskuudessa suosituin ketterä ohjelmistokehityksen menetelmä. VersionOne:n (2019) mukaan 72% yrityksistä käyttää Scrumia tai siihen pohjautuvia menetelmiä. Lisäksi Opelt et al. (2013, p. 6) väittävät, että scrum on täydellinen esimerkki ketterästä ohjelmistokehityksestä, sillä se noudattaa tiiviisti ketteryyden arvoja ja periaatteita keskittyessään tiimityöskentelyyn ja yhteistyöhön asiakkaan kanssa.

2.2 Ketterän ohjelmistokehityksen menetelmä: Scrum

Scrumin luoneet Shwaber & Sutherland (2017) määrittelevät scrumin *viitekehykseksi, jonka avulla voidaan käsitellä monimutkaisia ongelmia tuottaen samalla tuottavasti ja luovasti tuotteita, joilla on mahdollisimman paljon arvoa.* Viitekehys rakentuu iteraatiovaiheiden, eli scrumin tapauksessa **sprinttien**, ympärille, kuten kuvassa 1 on esitetty.



Kuva 1. *Scrumin viitekehys (löyhästi mukaillen Ashraf & Aftab 2017)*

Sprintit ovat kuin pieniä projekteja: ne tähtäävät ennalta määritettyyn päämäärään ja ovat ajallisesti, kustannuksiltaan ja laajuudeltaan rajattuja ainutlaatuisia kokonaisuuksia (Artto et al. 2008, s. 26). Sprintit ovat kuitenkin helpompia hallita kuin isot projektit: sprinttien lyhyt kesto, yleensä enintään kuukauden, parantaa ennustettavuutta ja pienentää riskejä, sillä suunnitelmia mukautetaan sprinttien välissä ja kustannukset toteutuvat kuukausittain. Lopullinen tuote syntyy usean sprintin **tuoteversion** summana. Myös scrumtiimi on oleellinen osa scrumia. Se koostuu scrummasterista, tuoteomistajasta ja 4-8 hengen kehitystiimistä. (Schwaber & Sutherland 2017).

Tuotteen kehitysjono on järjestetty lista tuotteeseen tarvittavista ominaisuuksista, sekä tuotteen vaatimuksista ja muutoksista. Tuotteen kehitysjono muuttuu jatkuvasti tuotteen muuttuvien tarpeiden mukaan. **Tuoteomistaja** varmistaa, että se pysyy ajan tasalla, ja että tuote saavuttaa mahdollisimman suuren arvon ja vastaa tehtyjä taloudellisia investointeja. (Schwaber & Sutherland 2017; Opelt et al. 2013, p. 16). **Sprintin suunnittelupalaverissa** scrumtiimi valitsee tuotteen kehitysjonosta ominaisuudet, jotka toteutetaan seuraavan sprintin aikana. Nämä ominaisuudet kerätään **sprintin kehitysjonoon**, jonka perusteella **kehitystiimi** suunnittelee ominaisuuksien toteutuksen ja toteuttaa ne sprintin aikana. Kehitystiimillä on päivittäin lyhyt **päiväpalaveri**, jossa he käyvät läpi, mitä edellisenä päivänä on tehty, mitä tehdään seuraavaan päiväpalaveriin mennessä ja mitä haasteita on tullut vastaan. Päiväpalaverin tarkoitus on parantaa kommunikointia tiimin sisällä, lisätä tietoisuutta sprintin tilanteesta ja ratkaista vastaan tulevat haasteet mahdollisimman nopeasti. **Scrummaster** auttaa kehitystiimiä ongelmien ratkaisussa, sekä varmistaa, että kaikki scrumtiimissä ymmärtävät scrumin toimintatavat ja toimivat niiden mukaan. (Schwaber & Sutherland 2017).

Toteutetut ominaisuudet testataan, ja kun kaikki ominaisuudet läpäisevät testit, voidaan sprintin tuoteversio asettaa **valmiiksi** (Ashraf & Aftab 2017). Schwaber & Sutherland (2017) mukaan valmiin tuoteversion tulee olla käytettävä ja julkaisukelpoinen sellaisenaan, mutta sen määritelmä voi vaihdella eri organisaatioissa ja eri tiimeissä organisaation sisällä. Scrumtiimillä tulee aina olla yhtenäinen käsitys siitä, mitä valmis tuoteversio tarkoittaa, sillä sen määritelmä asettaa tuoteversiolle minimivaatimukset. (Schwaber & Sutherland 2017). Valmis tuoteversio esitellään **sprintin katselmoinnissa** scrumtiimille ja muille sidosryhmille, ja kerätään heiltä palautetta. Palautteen perusteella lisätään muutoksia tuotteen kehitysjonoon, jotka toteutetaan myöhemmissä sprinteissä. (Ashraf & Aftab 2017). Lisäksi katselmoinnissa käydään läpi mitä sprintin aikana tehtiin, mitä haasteita kohdattiin ja miten ne ratkaistiin (Schwaber & Sutherland 2017).

Viimeinen tapahtuma ennen seuraavan sprintin alkua on **sprintin retrospektiivi**, jossa scrumtiimi pyrkii tunnistamaan kehityskohteita tiimin työskentelyyn liittyen ja tekee suunnitelman toiminnan kehittämiselle seuraavaa sprinttiä varten. Retrospektiivin jälkeen seuraava sprintti voidaan taas aloittaa sprintin suunnittelusta. (Schwaber & Sutherland 2017)

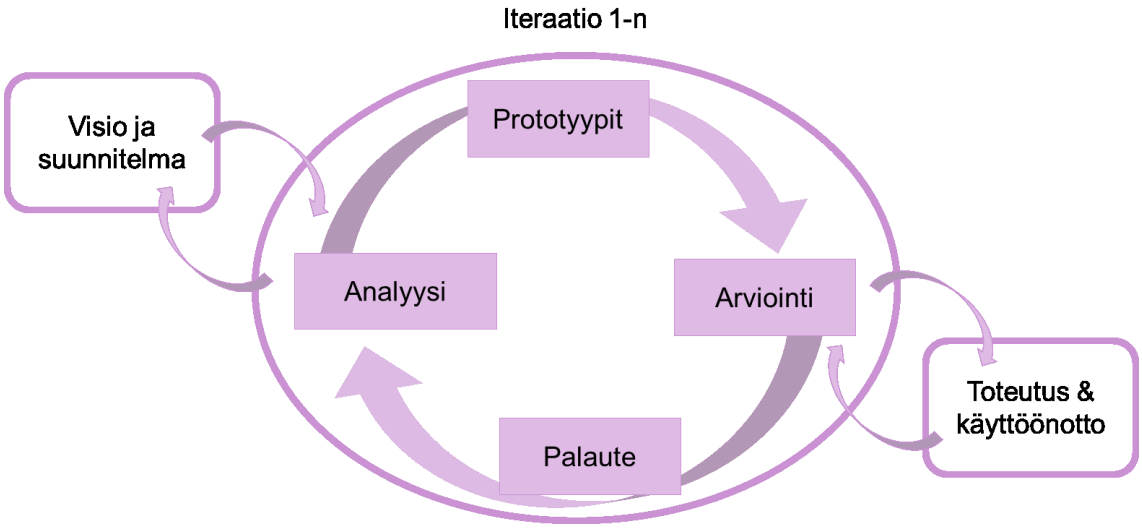
3. KÄYTTÄJÄKESKEISYYS KETTERÄSSÄ OHJELMISTOKEHITYKSESSÄ

3.1 Käyttäjäkeskeinen ohjelmistosuunnittelu

Käyttäjäkeskeisessä ohjelmistosuunnittelussa (*engl. user-centred systems design, UCSD*) käytettävyydestä tehdään ohjelmistokehityksen lähtökohta, sillä usein ohjelmistokehityksessä käyttäliittymä syntyy niin sanotusti ohjelmiston sivutuotteena (Gulliksen et al. 2003). Gulliksen et al. (2003) muodostivat aiemman kirjallisuuden pohjalta määritelmän käyttäjäkeskeiselle ohjelmistosuunnittelulle ja myöhemmin Lárusdóttir et al. (2017) ovat täsmentäneet määritelmää. Heidän mukaansa *käyttäjäkeskeinen ohjelmistosuunnittelu on prosessi, joka keskittyy käytettävyyteen ja käyttäjäkokemukseen ohjelmiston koko elinkaaren ajan, korostaen käyttöympäristön ymmärryksen, iteratiivisten prototyyppien käytön ja aktiivisen käyttäjäyhteistyön tärkeyttä*. Ohjelmiston käytettävyys muodostuu sen kyvystä mahdollistaa käyttäjille määriteltyjen tavoitteiden saavuttaminen tietyssä tilanteessa. Mitä pienemmillä resursseilla, tarkasti ja tyytyväisesti käyttäjä saavuttaa tavoitteensa, sitä parempi ohjelmiston käytettävyys on. (ISO 9241-11 1998). Käyttäjäkokemus alkaa muodostua jo ennen ohjelmiston käyttöä, jatkuen sen aikana ja vielä käytön jälkeenkin. Se muodostuu käyttäjän tunteiden, uskomusten, mieltymysten, reaktioiden, käytöksen ja saavutusten pohjalta, ja siihen vaikuttaa ohjelmiston ja käyttäjän lisäksi myös käyttöympäristö. (ISO 9241-210 2009).

Käyttäjäkeskeiseen ohjelmistosuunnitteluun osallistuva projektitiimi koostuu monialaisista ammattilaista (ISO 13407 1999) ja on hyvin tärkeää, että siihen sisältyy myös käytettävyyden asiantuntija suunnittelun alusta asti (Kapor 1990). Kaikkien projektiin osallistuvien tulee ymmärtää, ketä käyttäjät ovat, millainen on heidän käyttöympäristönsä, mikä on heidän toimintansa tarkoitus, miksi ja miten he hoitavat tehtävänsä ja niin edelleen. Kun käyttäjien tarpeet ymmärretään, käyttäjiin keskittyminen on helpompaa ja voidaan välttää keskittymisen siirtyminen teknisiin piirteisiin. (Gulliksen et al. 2003). Käyttäjien tavoitteiden, tehtävien ja tarpeiden tulee siis ohjata ohjelmistokehitystä alusta alkaen (ISO 13407 1999; Gould et al. 1997) ja heitä tulee osallistaa kehitysprosessiin aktiivisesti ja jatkuvasti koko ohjelmiston elinkaaren ajan (ISO 13407 1999; Gould et al. 1997; Nielsen). Tärkeää on myös se, että asiakasorganisaatioon sekä kehittäjäorganisaatioon saadaan luotua käyttäjäkeskeinen asenne (Gulliksen et al. 2003).

Kuvassa 2 on havainnollistettu käyttäjäkeskeisen ohjelmistosuunnittelun prosessi iteraatiovaiheineen. Suunnitteluprosessin alussa asetetaan käytettävyystavoitteet ja suunnittelun kriteerit, jotka ohjaavat kehitystä (Gould et al. 1997; Nielsen 1993). Kun karkea suunnitelma ja visio ovat valmiita, aloitetaan käyttäjien analyysi, eli hankitaan tietoa käyttäjistä, heidän käyttöympäristöstään, toiminnastaan ja tarpeistaan sekä käytettävyysvaatimuksista ja -tavoitteista (Gulliksen et al. 2003; Fox et al. 2008). Analyysin pohjalta luodaan yksityiskohtaisempi suunnitelma aloittaen yleiseltä tasolta ja edeten iteraatio iteraatiolta yksityiskohtaisempiin asioihin (Guliksen et al. 2003).



Kuva 2. *Käyttäjäkeskeinen ohjelmistosuunnittelu (mukaillen Gulliksen et al. 2003)*

Suunnitelmat tulee esittää mahdollisimman yksinkertaisesti ja ymmärrettävästi (Kyng 1995), sekä konkreettisessa muodossa, esimerkiksi prototyyppeinä tai luonnoksina (Gulliksen et al. 2003). Prototyyppejä käytetään hyödyksi arviointivaiheessa, jossa käyttäjät suorittavat niiden avulla työtehtäviään arvioidakseen suunnitelman sopivuutta käyttötarkoitukseensa (Cooper 1999). Käyttäjät siis arvioivat suunnitellut ratkaisut ennen kuin niitä viedään toteutukseen. Käyttäjien palautteen perusteella tehdään tarvittaessa muutoksia suunnitelmiin ja ne toteutetaan seuraavissa iteraatiovaiheissa. Yksi iteraatiovaihe voi olla kestoltaan vain puoli tuntia, kunhan se sisältää kaikki edellä luetellut vaiheet. (Gulliksen et al. 2003)

Kuten kuvasta 2 on nähtävissä, iteraatioiden suunnitelmaversiot siirtyvät valmistuessaan toteutukseen ja käyttöönottoon huolellisen arvioinnin jälkeen. Jokaisen iteraation jälkeen voidaan myös päivittää visiota ja suunnitelmaa, kun käyttäjistä ja heidän tarpeistaan on saatu lisää tietoa.

3.2 Käyttäjäkeskeisen ohjelmistosuunnittelun ja ketterän ohjelmistokehityksen yhteensovittaminen

Ketterissä ohjelmistokehityksen menetelmissä ei huomioida käytettävyyttä tai käyttökokemusta tarpeeksi (Armitage 2004; Lárusdóttir et al. 2017), eikä asiakkaan toiveiden perusteella rakennettu ohjelmisto ole välttämättä käytettävä (Fox et al. 2008). Käytettävyyden suunnittelun laiminlyönti johtaa asiakkaiden ja käyttäjien pettymiseen, sillä huonot käyttäjäkokemukset huomataan vasta, kun ohjelmisto on otettu käyttöön asiakasorganisaatiossa (Kuusinen & Väänänen-Vainio-Mattila 2012). Da Silva et al. (2018) korostaa, että ohjelmiston käytettävyys voi määrittää, onko ohjelmisto menestynyt vai epäonnistunut.

Käyttäjäkeskeisen ohjelmistosuunnittelun ja ketterän ohjelmistokehityksen yhteensovittamisen tarkoituksena on ohjelmistojen käytettävyyden parantaminen ja laatukriteerien täyttäminen (Da Silva et al. 2012; Lárusdóttir et al. 2017). Käyttäjäkeskeinen ohjelmistosuunnittelu mahdollistaa, että ohjelmistokehittäjät ymmärtävät käyttäjien tarpeita, ja kuinka käyttäjien tavoitteita ja aktiviteetteja tuetaan parhaiten ohjelmiston avulla. Ymmärryksen kautta on mahdollista tuottaa paremmin käytettävää ohjelmistoa ja saavuttaa parempi käyttäjätyytyväisyys. (Salah et al. 2014). Sy (2007) huomauttaa, että myös käyttäjäkeskeinen suunnittelu hyötyy ketteryydestä: Ketterissä projekteissa kaikki pitää saada valmiiksi iteraation lopussa, joten myöskään suunnitelmat eivät voi jäädä keskeneräisiksi toisin kuin esimerkiksi vesiputoustyyppisissä projekteissa. Myös Budvig et al. (2009) ovat huomanneet, että ketterissä projekteissa käytettävyyden suunnittelijat pääsevät tiiviimpään yhteistyöhön ohjelmistokehittäjien kanssa, ja käytettävyysongelmat huomataan ja korjataan nopeammin.

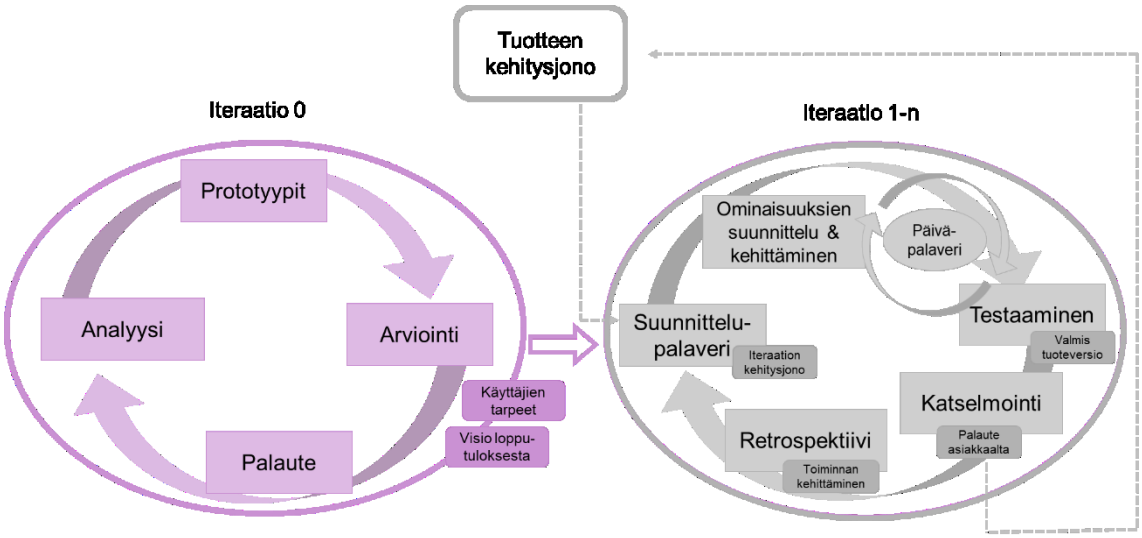
Seuraavissa alaluvuissa käsitellään, kuinka yhteensovittaminen voidaan toteuttaa. Alalukujen jaottelu pohjautuu Brhel et al. (2015) kokoamiin periaatteisiin käyttäjäkeskeisyyden ja ketteryyden yhteensovittamisesta, jotka he ovat tunnistaneet aiemman kirjallisuuden pohjalta. Periaatteet ovat seuraavat:

1. Projektin alussa on erillinen suunnitteluvaihe.
2. Tuote toteutetaan iteraatioiden ja tuoteversioiden pohjalta.
3. Käyttäjäkeskeinen suunnittelu ja ohjelmistokehitys toteutetaan erillisissä ja erivaiheisissa iteraatioissa, jotka kuitenkin nivoutuvat yhteen.
4. Sidosryhmiä osallistetaan tuotteen kehitykseen aktiivisesti.
5. Tuotosten kommunikointi sidosryhmille helposti ymmärrettävässä ja konkreettisessa muodossa.

Periaatteiden sekä scrumin ja käyttäjäkeskeisen ohjelmistosuunnittelun viitekehysten pohjalta rakennetaan uusi viitekehys pala palalta. Kuvissa 3-7 periaatteiden mukaan lisätyt ominaisuudet ovat värillisiä ja edellisestä kuvasta muuttumattomat ominaisuudet värittömiä. Aiemmista käyttäjäkeskeisen ketterän ohjelmistokehityksen viitekehyksistä (Miller 2005; Sy 2007; Fox et al. 2008) poiketen tämä viitekehys rakennetaan myös sellaisiin projekteihin sopivaksi, joihin ei ole mahdollista resursoida kahta erillistä tiimiä.

3.2.1 Suunnitteluvaihe: Iteraatio 0

Ensimmäisellä periaatteellaan Brhel et al. (2015) ottavat kantaa käyttäjäkeskeisyyden ja ketterän ohjelmistokehityksen väliseen ristiriitaan suunnittelusta: ketterässä ohjelmistokehityksessä pyritään välttämään liiallista suunnittelua, mutta käyttäjäkeskeisessä ohjelmistosuunnittelussa pidetään tärkeänä kattavaa suunnittelua ja analyysiä etenkin projektin alussa (Da Silva et al. 2018). Ketterässä ohjelmistokehityksessä suunnitellaan mahdollisimman vähän etukäteen, koska tavoitteena on säilyttää kyky mukautua muuttuviin vaatimuksiin (Ferreira et al. 2007a). Ensimmäisen periaatteen mukaan erillisen suunnitteluvaiheen lisääminen projektin alkuun, ennen tuotekehityksen aloittamista, auttaa ratkaisemaan suunnittelun ristiriidan. Esimerkiksi Sy (2007) ja Miller (2005) ehdottavat suunnittelun toteuttamista iteraatiossa 0 (kuva 3), jossa kerätään vaatimuksia, ymmärrystä käyttäjistä, heidän tavoitteistaan ja käyttöympäristöstään, sekä suunnitellaan käytettävyyttä (Miller 2005; Salah et al. 2014). Iteraation 0 avulla on mahdollista välttää kallis uudelleensuunnittelu, kun käytettävyysongelmat huomataan aiemmin ja vältetään huonojen suunnittelupäätösten tekeminen (Ferreira et al. 2007b). Iteraation 0 hyödyistä huolimatta, kaikkea ei kannata suunnitella etukäteen (Da Silva et al. 2012) ja sille tulisi asettaa aikaraja (Sy 2007). Esimerkiksi Fox et al. (2008) tapaustutkimuksessa iteraation 0 kesto vaihteli muutamasta päivästä neljään viikkoon eri yrityksissä.



Kuva 3. *Iteraatio 0: suunnitteluvaihe ennen tuotekehityksen aloittamista*

Iteraatiossa 0 kannattaa hyödyntää käytettävyyden suunnittelijoita, sillä heillä on paremmat valmiudet käytettävyysdatan keräämiseen ja analysointiin (Da Silva et al. 2012). Ohjelmistokehittäjät tarttuvat usein siihen, mitä asiakkaat tai käyttäjät haluavat, käyttäjien havainnoinnin sijaan (Peixoto & Da Silva 2009). Käytännössä käytettävyyden suunnittelijat keräävät käyttäjistä tietoa haastattelemalla, havainnoimalla ja keskustelemalla käyttäjien kanssa. Tietojen pohjalta voidaan muodostaa käyttäjätarinoita, persoonakuvauksia ja skenaarioita (Sy 2007), sekä matalankynnyksen prototyyppejä, kuten käsin piirrettyjä luonnoksia, muistilapuista koottuja malleja tai muita vastaavia havainnollistuksia. (Fox et al. 2008; Da Silva et al. 2012). Käyttäjäkeskeisen ohjelmistosuunnittelun mukaisesti prototyyppejä käytetään hyödyksi, kun käyttäjät arvioivat suunnitelmia (Fox et al. 2008). Iteraation 0 päättyessä valmiille tuotteelle on saatu määriteltyä tavoitteet ja luotua visio lopputuloksesta (Sy 2007). Lisäksi käyttäjien tarpeet ja vaatimukset tulee olla listattuna. Iteraation 0 jälkeen pidetään suunnittelukokous, jossa päätetään, mitkä ominaisuudet valitaan ensimmäiseen iteraatioon. (Fox et al. 2008).

3.2.2 Iteratiivisuus ja tuoteversiot

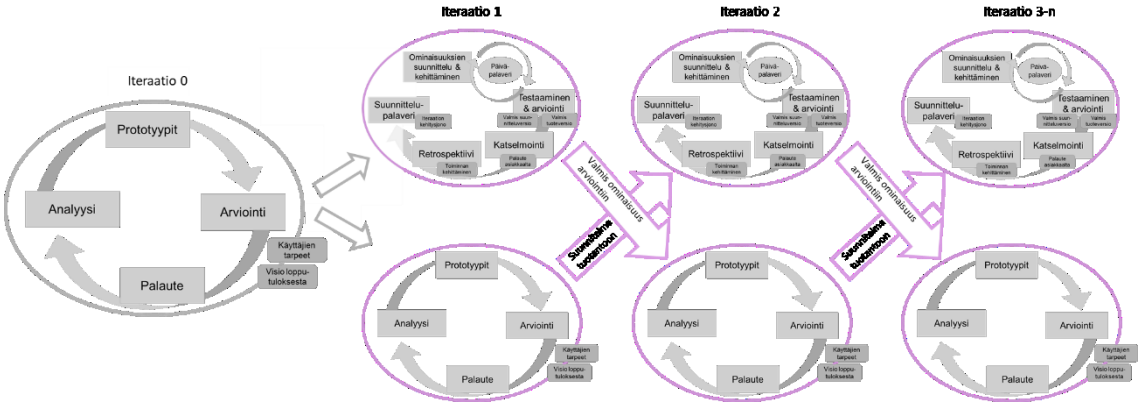
Toinen yhteensovittamisen periaate nostaa esiin iteroinnin ja tuoteversioiden julkaisemisen tärkeyden myös käyttäjäkeskeisessä ketterässä ohjelmistokehityksessä. Sekä käyttäjäkeskeisen suunnittelun että ketterän ohjelmistokehityksen kannalta on tärkeää, että tuoteversiosta saadaan palautetta usein ja tuoteversiota kehitetään palautteen perusteella. (Brhel et al. 2015). Ketterän ohjelmistokehityksen ja käyttäjäkeskeisen ohjelmistosuunnittelun iteratiivisuudessa on kuitenkin eroja, jotka asettavat haasteita yhteenso-

vittamiselle. Käyttäjäkeskeisessä ohjelmistosuunnittelussa iteraatioita käytetään havaittujen ongelmien korjaamiseen (Fox et al. 2008), kun taas ketterässä ohjelmistokehityksessä iteraatioon valitaan joukko ominaisuuksia, joista valmis tuote rakentuu vähitellen iteraatio kerrallaan. Ohjelmiston kehittäminen ominaisuus kerrallaan vaikeuttaa suunnittelijoiden kokonaiskuvan hahmottamista (Salah et al. 2014).

Käyttäjäkeskeisessä ketterässä ohjelmistokehityksessä myös suunnittelu tulee paloitella ja luoda valmis suunnitelma suunnitteluversioiden pohjalta (kuva 5) – kuten ketterässä ohjelmistokehityksessä luodaan valmis tuote tuoteversioiden pohjalta – jolloin tavoitteena on saavuttaa suunnittelun tavoitteet vähitellen (Sy 2007). On kuitenkin haastavaa määrittää, minkä kokoinen suunnittelun palanen sopii tiettyyn iteraatioon ja kuinka paljon suunnittelua yhdessä iteraatiossa tulisi tehdä (Tzanidou & Ferreira 2010). Suunnittelun paloittelu kannattaakin hoitaa määrittelemällä suunnittelun tavoitteet todella hyvin (Sy 2007), paloittelemalla suunnittelu iteraatioon valittujen ominaisuuksien mukaan (Nafaji & Toyoshiba 2008) ja asettamalla käyttäjäkeskeiselle suunnittelulle aikarajat (Hodgetts 2005). Haasteista huolimatta Sy (2007) näkee myös hyötyjä suunnittelun paloittelussa: Koska ketterissä projekteissa keskitytään vain muutamaan ominaisuuteen kerrallaan, ei suunnittelijoidenkaan tarvitse tehdä kaikkea suunnittelutyötä kerralla.

3.2.3 Irralliset iteraatiovaiheet ohjelmistokehitykselle ja käyttäjäkeskeiselle ohjelmistosuunnittelulle

Kolmannen periaatteen (Brhel et al. 2015) mukaan käyttäjäkeskeinen suunnittelu ja ohjelmistokehitys tulisi toteuttaa vierekkäisissä, yhteen nivoutuneissa vaiheissa (kuva 4). Esimerkiksi Sy (2007) ja Miller (2005) ehdottavat, että käyttäjäkeskeinen suunnittelu tehdään seuraavaa iteraatiota varten edellisen iteraation aikana, jolloin suunnitelmat ovat valmiina ennen kuin ohjelmistokehitys alkaa seuraavassa iteraatiossa. Lisäksi suunnitelmat tulee olla arvioituja käyttäjien toimesta, sillä suunnitelmien muuttaminen on helpompaa, kun toteutusta ei ole vielä aloitettu. Irralliset iteraatiovaiheet jatkuvat, kunnes tuote on valmis (Sy 2007).



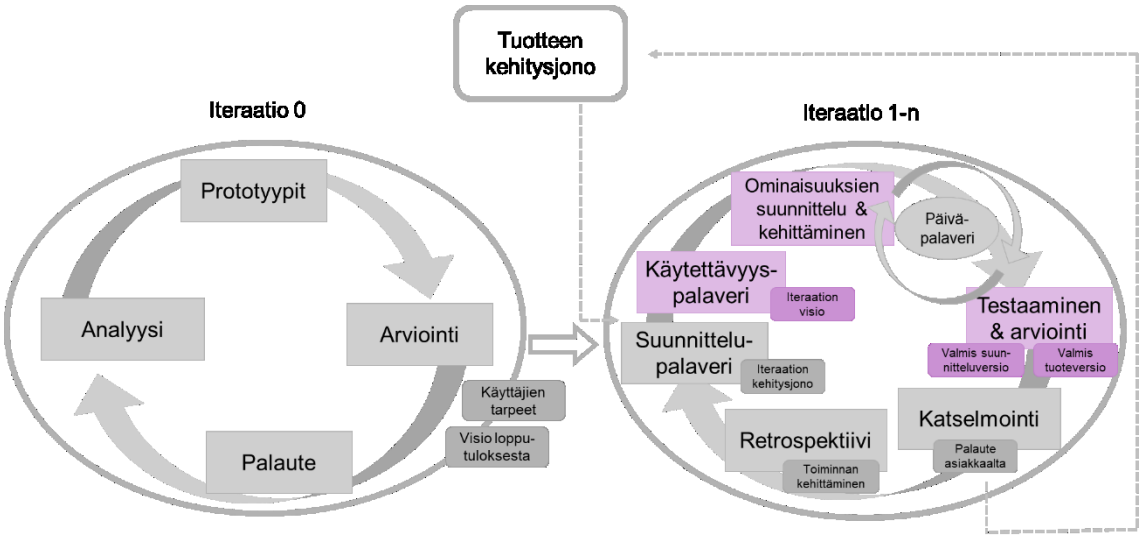
Kuva 4. *Irralliset iteraatiovaiheet ohjelmistokehitykselle ja -suunnittelulle (mukaillen Sy 2007)*

Käyttäjäkeskeisen ohjelmistosuunnittelun iteraatioissa toistetaan samoja aktiviteetteja kuin iteraatiossa 0 (Da Silva et al. 2012). Ennen ohjelmistokehityksen iteraatiota, suunnittelijoiden tulee saada suunnitelmat valmiiksi siihen valittujen ominaisuuksien osalta. Kun ohjelmistokehittäjät ovat toteuttaneet ominaisuudet, suunnittelijat antavat ne käyttäjien arvioitavaksi seuraavassa iteraatiossa. Suunnittelijat toteuttavat siis saman iteraation aikana seuraavan ohjelmistokehittäjien iteraation suunnitelmat ja edellisen ohjelmistokehittäjien iteraation ominaisuuksien arvioinnit. (Sy 2007; Fox et al. 2008). Kuitenkin iteraatio yksi eroaa hieman seuraavista iteraatioista sekä ohjelmistokehityksen että suunnittelun osalta. Ohjelmistokehityksen ensimmäisessä iteraatiossa tulisi keskittyä ominaisuuksiin, joilla on korkeat kehityskustannukset, mutta matalat suunnittelukustannukset (Miller 2005; Sy 2007). Suunnittelijoiden ensimmäisessä iteraatiossa jatketaan käyttäjien tarpeiden määrittelyä seuraavaa iteraatiota varten, mutta koska ominaisuuksia ei ole vielä valmistunut, valmiiden ominaisuuksien arviointia ei tarvitse vielä tehdä (Sy 2007).

Ketterässä ohjelmistokehityksessä korostetaan, että kehitystiimistä tulee löytyä kaikki tarvittava osaaminen asetettujen tavoitteiden saavuttamiseksi. Mikäli suunnittelua kuitenkin toteutetaan samaan aikaan kuin kehitystä, mutta yksi vaihe sitä edellä, kahden erillisen tiimin kokoaminen on tarpeellista (Brhel et al. 2015). Budvig et al. (2009) ehdottavat jopa käytettävyydestä vastaavan tuoteomistajan nimittämistä scrum-projekteissa jo valmiiksi olevien tuoteomistajien lisäksi.

Brhel et al. (2015) huomauttavat, että tehtävien suorittaminen päällekkäin on haastavaa toteuttaa käytännössä. Myös Lárusdóttir et al. (2017) mukaan yksi suurimmista haasteista ketterän ohjelmistokehityksen ja käyttäjäkeskeisen suunnittelun yhteensovittamisessa on tasapainoilu ohjelmistokehittäjien ja suunnittelijoiden iteraatioiden välillä. Myös

Miller (2005) myöntää, ettei kahden irrallisen iteraatiovaiheen toteuttaminen ole mahdollista kaikissa organisaatioissa, mutta huomauttaa kuitenkin, että irralliset iteraatiovaiheet lisäävät suunnittelijoiden ja ohjelmistokehittäjien välistä yhteistyötä, pienentävät suunnittelun ja ohjelmistokehityksen kustannuksia ja mahdollistavat laadukkaampien tuotteiden tuottamisen. Lisäksi Budvig et al. (2009) mukaan samassa iteraatiossa toteutettava ohjelmistokehitys ja suunnittelu eivät toimi syntyvistä kommunikaatio-ongelmista ja vaatimusten huonosta dokumentoinnista johtuen. Siis kahden tiimin kokoaminen on suositeltavaa, mutta mikäli se ei ole mahdollista, tämän periaatteen mukaisia muutoksia ei voida ottaa käyttöön. Sen sijaan, käytännössä tulee testata suunnittelu- ja kehittämisaktiviteettien toteuttamista peräkkäin ja limittäin (kuva 6), sekä pyrkiä välttämään Budvig et al. (2009) havaitsemat kommunikaatio-ongelmat, mihin palataan alaluvussa 3.2.5.



Kuva 5. *Käyttäjäkeskeisen ohjelmistosuunnittelun ja ketterän ohjelmistokehityksen aktiviteetit limittäin*

Kuvassa 5 havainnollistetaan vaihtoehtoista viitekehystä, jossa käyttäjäkeskeisen ohjelmistosuunnittelun ja ketterän ohjelmistokehityksen aktiviteetit toteutetaan limittäin. Viitekehykseen on lisätty käytettävyyspalaveri, jossa sprinttiin valitut ominaisuudet käydään läpi yhdessä käyttäjien kanssa. Käytettävyyspalaverissa tehdään iteraation ensimmäiset käyttäjäkeskeisen suunnittelun iteraatiot, iteraation vision selvittämiseksi. Budvig et al. (2009) ehdottavat, että käytettävyyden suunnittelulle olisi oma käytettävyyden kehitysjono. Tässä viitekehyksessä iteraation visio vastaa edotettua käytettävyyden kehitysjonoa, mutta se on huomattavasti kevyempi tapa dokumentoida käytettävyyden tavoitteita iteraation ajalle. Visio pohjautuu iteraation kehitysjonoon, ja vision voi käytännössä dokumentoida iteraation kehitysjonoon tai prototyyppeinä kuvaamaan iteraation tavoiteltua lopputulosta.

Kuvan 5 ”Ominaisuuksien suunnittelu ja kehittäminen” käsittää myös käytettävyyssuunnittelun toisin kuin scrumin viitekehyksessä kuvassa 1, jossa samalla termillä tarkoitetaan pelkästään ominaisuuksien teknistä suunnittelua. Testaaminen ja arviointi ovat erotettu erillisiksi toimenpiteiksi: testaamisella tarkoitetaan sisäistä ja mekaanista ominaisuuksien testaamista, kun taas arvioinnilla tarkoitetaan käyttäjien hyödyntämistä ominaisuuksien käytettävyyden ja suunnitelmien arvioinnissa. Testaamisen ja arvioinnin tavoitteena on tuottaa iteraation aikana valmis suunnitteluversio ja edelleen valmis tuoteversio sprinttiin valituista ominaisuuksista. Suunnitteluversio voi valmistua joko huomattavasti ennen tuoteversiota, tai suunnitteluversiota voidaan kehittää ominaisuus ominaisuudelta rinnakkain tuoteversion kanssa. Valmis tuoteversio kuitenkin sisältää aina valmiin suunnitteluversion.

3.2.4 Sidosryhmien osallistaminen

Neljäs periaate korostaa sidosryhmien roolia käyttäjäkeskeisessä ketterässä ohjelmistokehityksessä: heitä tulee aktiivisesti osallistaa suunnitteluun ja kehitykseen alusta alkaen, koko projektin ajan (Brhel et al. 2015). Molemmat menetelmät ovat jo itsessään ihmiskeskeisiä, mutta käyttäjäkeskeisessä ohjelmistosuunnittelussa keskitytään nimensä mukaisesti käyttäjiin, kun taas ketterässä ohjelmistokehityksessä keskitytään asiakkaaseen (Fox et al. 2008). Arvon luominen asiakkaalle ei kuitenkaan takaa laadukasta, käytettävää ohjelmistoa (Lárusdóttir et al. 2017), jonka vuoksi käyttäjien huomioiminen on tärkeää.

Osallistamiseen tulee sisältyä ainakin ominaisuuksien arviointia loppukäyttäjien toimesta (Brhel et al. 2015). Kuitenkaan käytettävyyden arvioinnissa ei ole aina välttämätöntä käyttää oikeita käyttäjiä: Kun käyttäjiä ei ole saatavilla, esimerkiksi muut projektitiimin jäsenet voivat hoitaa arvioinnin (Lievesley & Yee 2007; Fox et al. 2008). Käyttäjien korvaaminen projektitiimin jäsenillä säästää aikaa ja kustannuksia, samoin kuin prototyyppien käyttäminen ja arvioinnin toteuttaminen etänä (Salah et al. 2014). Sy (2007) mukaan erityisesti aikaisimpien iteraatioiden tuoteversioiden käytettävyyttä ei kannata arvioida käyttäjillä, vaan heidän hyödyntämistään kannattaa säästellä keski- ja myöhäisen vaiheen iteraatioihin, jolloin arvioinnissa on tarkoitus selvittää ovatko aiemmin asetetut tavoitteet saatu täytettyä. Aikaisimmat tuoteversiot kannattaa antaa käyttäjien ”sijaisten” testattavaksi. Heidän tulee kuitenkin vastata taitotasoltaan oikeita käyttäjiä. (Sy 2007).

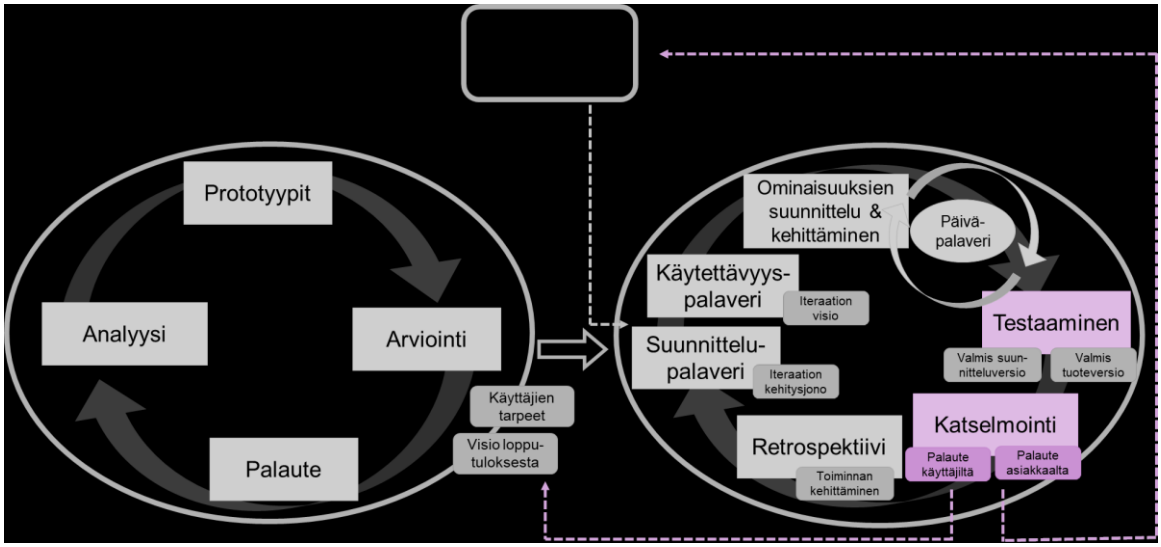
Beyer (2010) huomauttaa, että käyttäjien kannattaa arvioida ominaisuudet aina, kun se on mahdollista, sillä käyttäjien ”sijaiset” eivät tiedä, millaisia käyttäjien tarpeet ja aktiviteetit ovat. Lisäksi ongelmat löytyvät todennäköisesti aikaisemmin, jos oikeat käyttäjät

arvioivat ominaisuudet (Da Silva et al. 2012), ja ainoastaan käyttäjät voivat määritellä kuinka tyytyväisiä he ovat ominaisuuksiin ja kuinka arvokas tuote on heille (Lárusdóttir et al. 2017). Lárusdóttir et al. (2017) suosittelevat, että käyttäjien kanssa kommunikoidaan vähintään kerran jokaisen iteraation aikana ja ominaisuudet arvioidaan käyttäjillä vähintään joka toisessa iteraatiossa, mutta toisinaan ominaisuudet eivät ole valmiita käytettävyyden arviointiin iteraation jälkeen (Detweiler 2007).

Mikäli tapaamista käyttäjien kanssa ei voida sopia kasvotusten, prototyyppien jakaminen yhteisellä alustalla ja kommentointimahdollisuuden antaminen käyttäjille voi riittää (Lárusdóttir et al. 2017). Tuoteversioiden valmistumisen jälkeiset loppupalaverit tarjoavat hyvän mahdollisuuden kerätä palautetta myös käyttäjiltä (Kane 2003). Toisaalta jos testaaminen toteutetaan vasta iteraation lopussa ongelmien korjaamiseen ei jää tarpeeksi aikaa (Salah et al. 2014). Lárusdóttir et al. (2013) suosittelee arviointien tekemistä myös ennen ominaisuuksien toteutusta.

Kun käyttäjät ovat arvioineet ominaisuudet, arviointien tulokset tulee käydä koko projektitiimin kesken yhdessä läpi (Lárusdóttir et al. 2017). Jos ominaisuudet eivät läpäise käytettävyyden arviointia, ne palautetaan kehitystiimille korjattaviksi vielä saman iteraation aikana, jos se on mahdollista. Vasta kun ominaisuudet ovat läpäisseet käytettävyyden arvioinnin, niitä voidaan pitää valmiina. (Fox et al. 2008). Käyttäjien muutostoiveista tärkeimpiä ovat ne, jotka vaikuttavat ohjelmiston kilpailukykyyn käyttäjien näkökulmasta (Lárusdóttir et al. 2017). Ohjelmisto voi kilpailla esimerkiksi kilpailijoiden tuotteiden tai käyttäjien vanhojen menetelmien kanssa.

Käytettävyyden arviointi sisältää käyttäjien suorituksen mittaamista, käyttäjien suorittaessa huolella suunniteltuja tehtäviä (Salah et al. 2014). Perusteelliseen arviointiin ei kuitenkaan ole aina tarpeeksi aikaa, koska iteraatiot ovat lyhyitä (Lárusdóttir et al. 2017), joten nopeille matalankynnyksen tavoille kerätä palautetta käyttäjiltä on suuri tarve (Lárusdóttir et al. 2013). Lárusdóttir et al. (2013) mainitsevat, että arviointia voi toteuttaa hyödyntäen käyttäjätarinoiden kuvauksia tehtävien suunnittelussa, kutsumalla käyttäjiä kommentoimaan suunnitelmia tai teettämällä käyttäjille kyselyitä.



Kuva 6. *Sidosryhmien osallistaminen*

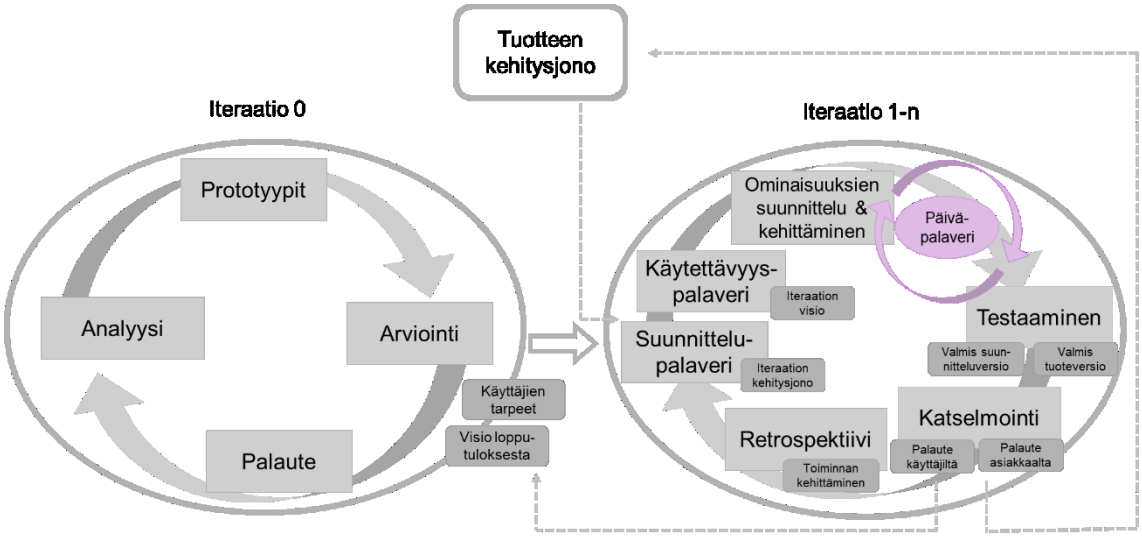
Käyttäjien käyttäminen arvioinneissa ei ole aina mahdollista, jonka vuoksi kuvan 6 viitekehyksessä arviointi on siirretty toteutettavaksi vasta katselmoinnissa. Katselmointiin kutsutaan projektitiimin lisäksi asiakas ja käyttäjiä, joilta pyydetään palautetta tuoteversiosta. Käyttäjiltä pyritään keräämään palautetta erityisesti käytettävyyteen liittyen, jolloin palaute vaikuttaa visioon lopputuloksesta, täsmentäen ja muokaten aiemmin luotuja prototyyppejä. Tuotteen kehitysjonoon tehdään muutoksia asiakkaan palautteen pohjalta. Myös käyttäjien toivomia ominaisuuksia voidaan lisätä tuotteen kehitysjonoon, jos asiakas hyväksyy muutokset. Ennen katselmointia käytettävyyden arviointi toteutetaan sisäisiä testaajia hyödyntäen, jonka vuoksi kuvassa 6 käytettävyyden arviointia ei nosteta esiin, vaan arviointi toteutetaan testaamisen ohessa.

3.2.5 Kommunikointi

Viidennen periaatteen mukaan suunnitelmista ja kehityksestä pitäisi kommunikoida sidosryhmien kanssa aineellisten ja näkyvien tuotosten, kuten prototyyppien tai rautalankamallien, kautta (Salah et al. 2014; Brhel et al. 2015). Erityisesti prototyypit sopivat moneen käyttötarkoitukseen: niiden avulla voidaan kerätä palautetta sidosryhmiltä, neuvotella eri suunnitteluratkaisujen paremmuudesta ja selittää tulevaisuuden käyttötilanteet käyttäjille mahdollisimman vaivattomasti (Da Silva et al. 2012; Lárusdóttir et al. 2017).

Vaikka Brhel et al. (2015) kokoamissa periaatteissa ei mainita projektitiimin välisen kommunikoinnin tärkeyttä, tiivis yhteistyö ohjelmistokehittäjien ja käytettävyyden suunnittelijoiden välillä on yksi suurimmista menestystekijöistä käyttäjäkeskeisen ohjelmistosuunnittelun ja ketterän ohjelmistokehityksen yhteensovittamisessa (Da Silva et al. 2011;

Kuusinen & Väänänen-Vainio-Mattila 2012). Tiiviillä kommunikoinnilla voidaan välttää viivästykset ja pullonkaulat (Ferreira et al. 2007), vähentää virallisten dokumenttien tarvetta suunnittellussa (Sy 2007) ja parantaa koko tiimin ymmärrystä käyttäjistä (Salah et al. 2014). Tiiviimmän yhteistyön saavuttamiseksi, Salah et al. (2014) ja Sy (2007) ehdottavat käytettävyyden suunnittelijoiden osallistumista päiväpalavereihin yhdessä ohjelmistokehittäjien kanssa.



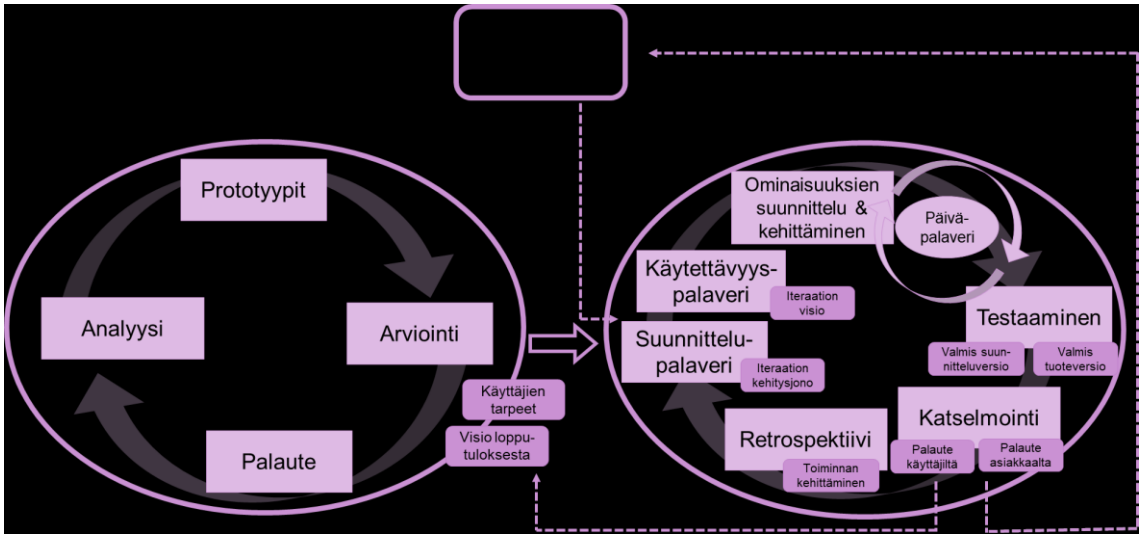
Kuva 7. *Käytettävyyden suunnittelijoiden osallistuminen päiväpalavereihin ohjelmistokehittäjien kanssa*

Kuvassa 7 ei ole näkyvää muutosta aiempiin viitekehyksen vaiheisiin. Tämän periaatteen pohjalta on kuitenkin perusteltua, että myös käytettävyyden suunnittelijat osallistuvat päiväpalavereihin.

3.2.6 Käyttäjäkeskeisen ketterän ohjelmistokehityksen viitekehys

Edellä esiteltyjen periaatteiden ja rajoitusten pohjalta koottu viitekehys (kuva 8) alkaa iteraatiosta 0, jossa kerätään tietoa käyttäjien tarpeista ja vaatimuksista sekä luodaan visio ohjelmiston lopputuloksesta. Käyttäjiä voidaan haastatella, havainnoida tai heille voidaan teettää kyselyitä työtehtävien, tarpeiden ja ominaisuuksien tärkeyden selvittämiseksi. Tarpeiden pohjalta luodaan matalankynnyksen prototyyppejä, esimerkiksi piirroksia, joiden avulla suunnitelmat välitetään käyttäjille helposti ymmärrettävässä ja konkreettisessa muodossa. Prototyypit mahdollistavat myös suunnitelmien arvioinnin, jolloin käyttäjät voivat antaa palautetta suunnitelmista jo projektin varhaisessa vaiheessa, kun

tuotantoa ei ole vielä aloitettu. Palautteen pohjalta suunnitelmia voidaan muokata ja lopulta muodostetaan suunnitelma, joka vastaa visiota lopputuloksesta. Visio lopputuloksesta dokumentoidaan prototyyppinä tai muulla konkreettisella tavalla. Vision ei ole tarkoitus olla yksityiskohtainen, vaan yksityiskohtia lisätään seuraavissa iteraatioissa. Iteraation 0 aikana voidaan myös hyödyntää käyttäjiä paremmin paikkansapitävien käyttäjätarinoiden ja myöhemmissä testausvaiheissa käytettävien skenaarioiden luomisessa. Iteraation 0 kesto voi vaihdella päivistä viikkoihin, mutta on tärkeää, että iteraation kestosta on sovittu etukäteen. Aikarajoituksella pyritään välttämään liiallista suunnittelua.



Kuva 8. *Viitekehys käyttäjäkeskeiselle ketterälle ohjelmistokehitykselle*

Seuraavissa iteraatioissa toteutetaan käyttäjäkeskeistä suunnittelua vain silloin, kun iteraatioon valitut ominaisuudet vaikuttavat ohjelmiston käytettävyyteen. Usein käyttäjäkeskeiset iteraatiot painottuvat projektin keski- ja loppupuolelle. Kuten muutkin iteraatiot, käyttäjäkeskeiset iteraatiot alkavat suunnittelupalaverilla, jossa päätetään asiakkaan kanssa, mitkä ominaisuudet valitaan kyseiseen iteraatioon. Suunnittelupalaverin jälkeen pidetään toinen palaveri käytettävyydestä käyttäjien kanssa. Käytettävyyspalaverissa kartoitetaan tarkemmin käyttäjien tarpeita ja käyttöskenaarioita valittuihin ominaisuuksiin liittyen haastatteluin, havainnoimalla tai kyselyllä. Tietojen pohjalta luodaan prototyyppi, esimerkiksi piirros tai rautalankamalli, jonka avulla käyttäjät voivat arvioida suunnitelman. Arvioinnin pohjalta suunnitelmaa voidaan muokata, kunnes suunnitelma on valmis iteraation visioksi. Käytettävyyspalaverin kesto voi vaihdella tunneista päiviin. Tärkeää on, että käytettävyyspalaverin kestosta on sovittu etukäteen.

Käytettävyyspalaverin jälkeen iteraatioon valittujen ominaisuuksien suunnittelua ja kehittämistä jatketaan, kunnes iteraatio tulee päätökseensä. Sekä muuttuneita käytettävyyssuunnitelmia että ominaisuuksien toiminnallisuuksia testataan ja arvioidaan sisäisten

testaajien toimesta, resurssien säästämiseksi. Testaajien kannattaa olla taidoiltaan käyttäjiä vastaavalla tasolla, sillä esimerkiksi ohjelmistokehittäjä ei todennäköisesti huomaa samoja käytettävyysongelmia kuin käyttäjä huomaisi. Paras testaaja on kuitenkin aina oikea käyttäjä, joten käyttäjät arvioivat valmistuneet ominaisuudet iteraation katselmoinnissa. Arviointia varten käyttäjälle annetaan oikeaa käyttötapausta vastaava tehtävä suoritettavaksi. Käyttäjää voidaan haastatella tai havainnoida, jotta saadaan kerättyä palautetta ominaisuuksista ja löydettyä mahdolliset ongelmat. Käyttäjän palautteen perusteella voidaan muuttaa ja tarkentaa visiota lopputuloksesta. Muutokset tuotteen kehitysjonoon tehdään aina asiakkaan hyväksynnän kautta, joten käyttäjän ehdottaessa uutta ominaisuutta, asiakkaan tulee hyväksyä se. Katselmoinnin jälkeen pidettävään retrospektiiviin ei tule muutoksia käyttäjäkeskeisyyden osalta.

Viitekehys (kuva 8) noudattaa lähes kaikkia aiemmin alaluvussa 3.2. listattuja periaatteita käyttäjäkeskeisyyden ja ketteryyden yhteensovittamisesta (Brehl et al. 2015). Vain kolmas periaate ”Käyttäjäkeskeinen suunnittelu ja ohjelmistokehitys toteutetaan erillisissä ja erivaiheisissa iteraatioissa, jotka kuitenkin nivoutuvat yhteen” ei päde viitekehykseen, sillä erillisten ja erivaiheisten iteraatioiden toteuttaminen yhdellä tiimillä ei olisi mahdollista. Tämän viitekehyksen tapauksessa kolmas periaate voisi olla ”Iteraatioihin lisätään käyttäjäkeskeisyyttä järjestämällä käytettävyyspalavereita ja kutsumalla käyttäjät katselmointeihin”. Seuraavaksi empiirisessä tutkimusvaiheessa arvioidaan viitekehykseen tehtyjä muutoksia: käyttäjäkeskeisyyden ja ketteryyden yhdistämistä hyödyntäen vain yhtä tiimiä, käytettävyyspalavereiden lisäämistä iteraatioihin ja käyttäjien kutsumista katselmuksiin. Lisäksi tutkimuksessa selvitetään viitekehyksen sopivuutta projekteihin, joissa valmis alusta asettaa rajoituksia käyttäjien toiveiden toteuttamiselle.

4. TUTKIMUKSEN TOTEUTUS

Tämä tutkimus toteutetaan tapaustutkimuksena, jossa kohdeyrityksen henkilöstöä haastatellaan tavoitteena arvioida luvussa 3 kootun viitekehyksen toimivuutta kohdeyrityksen projekteissa. Tapaustutkimuksessa on mahdollista testata valmista teoriaa sen luonnollisessa käyttöympäristössä (Saunders et al. 2009, s. 147), minkä vuoksi se sopii tutkimusstrategiaksi tähän tutkimukseen ja sen tavoitteisiin. Aineisto kerätään semi-strukturoiduin haastatteluin, sillä haastattelut mahdollistavat lisäkysymysten kysymisen ja haastatteluiden mukauttamisen haastateltavan ja tilanteen mukaan (Saunders et al. 2009, s. 324).

Tutkimuksen kohdeyritys on suuri ohjelmistoalan yritys, joka toimii Suomessa ja kansainvälisesti. Haastatteluihin pyydettiin henkilöitä vain yhdestä, noin 100 henkilön yksiköstä, joka keskittyy Microsoft Dynamics 365 tuotteiden myyntiin ja kustomointiin. Vaikka tutkimuksen kohteena onkin vain yksi yksikkö, tutkimuksessa tutkitaan useita tapauksia, sillä haastateltavat henkilöt työskentelevät useissa keskenään hyvin erilaisissa projekteissa. Kohdeyksikön projekteista suurin osa toteutetaan ketterän ohjelmistokehityksen mukaisesti, joten projektien erot syntyvät etenkin projektien kokoluokista ja erilaisista tavoista ottaa käyttäjät huomioon projekteissa. Aiempaan kirjallisuuteen verraten kohdeyksikön projektit poikkeavat aiemmista tutkituista projekteista, sillä kohdeyksikössä ei ole erillistä käytettävyystiimiä tai käytettävyyden ammattilaisia. Lisäksi Microsoft Dynamics 365 tuotteet asettavat kohdeyksikölle rajoitteita käytettävyyden suunnittelussa, sillä kohdeyksikkö ei voi vaikuttaa kaikkiin ohjelmiston ominaisuuksiin. Aiemmassa kirjallisuudessa ohjelmistokehityksen kohteena on ollut yritysten omat ohjelmistotuotteet, joko alkukehityksestä lähtien tai jatkokehityksen kohteena.

Kohdeyksiköstä haastateltavat henkilöt valittiin harkintaan pohjautuvan otannan mukaisesti. Tavoitteena oli kerätä haastateltavia eri työtehtävistä ja eri projekteista, jotta haastattelussa saataisiin kerättyä mahdollisimman monipuolisia vastauksia. Haastatellut henkilöt on koottu taulukkoon 1. Puolet haastatelluista ovat olleet mukana projekteissa, joissa loppukäyttäjiä osallistettiin ohjelmiston suunnitteluun tai arviointiin projektin aikana tai ohjelmiston jatkokehityksessä käyttöönoton jälkeen.

Taulukko 1. *Haastateltavien työtehtävät ja loppukäyttäjien osallistaminen projekteissa*

Haastateltava	Työtehtävä	Loppukäyttäjiä projekteissa
H1	Tuoteasiantuntija	Ei
H2	Projektipäällikkö	Ei
H3	Projektipäällikkö	Kyllä
H4	Tuoteasiantuntija	Kyllä
H5	Projektipäällikkö	Kyllä
H6	Integraatiopäällikkö	Ei
H7	Sovelluskehittäjä	Ei
H8	IT-Arkkitehti	Ei
H9	Ohjelmistokehittäjä	Kyllä
H10	Ohjelmistokehittäjä	Kyllä

Haastattelut toteutettiin ensisijaisesti kasvotusten, mutta koska tapaaminen ei ollut mahdollista kaikkien haastateltavien kanssa, osa haastatteluista toteutettiin Teams-sovelluksen kautta. Sovelluksen avulla oli mahdollista jakaa haastatteluun liittyvä materiaali reaaliaikaisesti. Haastattelun aikana haastateltaville esiteltiin yksinkertaistettu viitekehys, jossa painotettiin analyysivaihetta, käytettävyyspalaveria ja katselmointia, minkä jälkeen heille esitettiin seuraavan haastattelurungon mukaisia kysymyksiä:

1. Miten viitekehyksen käyttöönotto näkyisi työnkuvassasi? Olisiko muutos positiivinen vai negatiivinen?
2. Mitä ongelmia tai haasteita viitekehyksen käyttöönotosta voisi seurata?
3. Mitä ongelmia viitekehyksen käyttöönotto voisi ratkaista? Mitkä viitekehyksen osa-alueet olisivat osa ratkaisua?
4. Kokisitko mahdollisena noudattaa viitekehystä yhden tiimin voimin vai tarvittaisiinko siihen toinen tiimi?
5. Miten muuttaisit viitekehystä?

Lisäksi haastateltavien vastausten pohjalta esitettiin tarkentavia lisäkysymyksiä paremman ymmärryksen saamiseksi ja jotta vastaukset voitiin kohdistaa tiettyyn viitekehyksen osa-alueeseen. Semi-strukturoidulle haastattelulle tyypilliseen tapaan (Saunders et al. 2009, s. 320), joitakin kysymyksiä painotettiin enemmän toisissa haastatteluissa. Esimerkiksi haastatteluissa, joissa haastateltavat ovat olleet mukana projekteissa, joissa loppukäyttäjiä on osallistettu projektin aikana, painotettiin erityisesti jo havaittujen hyötyjen ja haasteiden selvittämistä kysellen aiemmista kokemuksista käyttäjien osallistamiseen liittyen.

Haastattelurungon ensimmäisen kysymyksen tarkoitus on johdatella haastateltava kuvittelemaan viitekehksen noudattaminen käytännössä, heidän oman työnkuvansa kautta, jolloin muutoksen aiheuttamat hyödyt ja haitat voivat olla helpompi hahmottaa. Seuraavassa kysymyksessä annetaan haastateltavalle mahdollisuus miettiä haasteita laajemmin kuin oman työnkuvansa kannalta. Toisen kysymyksen avulla varmistetaan myös, että haasteet tulevat mainituksi, mikäli oman työnkuvan kannalta viitekehyksen käyttöönotto vaikuttaa oman työtehtävän kannalta positiiviselta. Kolmannessa kysymyksessä selvitetään viitekehyksen hyötyjä olemassa olevien ongelmien kautta, jotta hyödyt olisivat mahdollisimman käytännönläheisiä. Neljännessä kysymyksessä selvitetään suoraviivaisesti, voiko viitekehystä noudattaa yhdellä tiimillä. Saman kysymyksen aikana selvitetään myös kumpi on parempi vaihtoehto, yksi tiimi vai kaksi tiimiä. Viimeisen kysymyksen aikana annettiin haastateltavalle mahdollisuus kertoa kehitysehdotuksia: mitä hän muuttaisi viitekehyksessä, onko siinä jotain ylimääräistä tai pitäisikö siihen lisätä jotakin.

Haastattelun aikana painotettiin haastateltaville, että kaikenlainen palaute viitekehyksestä, myös kritiikki, on toivottavaa ja että myös viitekehyksen esittämisen aikana saa keskeyttää, mikäli viitekehyksestä tulee mieleen kysymyksiä tai kommentteja. Haastatteluiden kesto vaihteli 25 ja 45 minuutin välillä ja haastattelut tallennettiin ensisijaisesti äänittämällä ja toissijaisesti kirjoittamalla, mikäli äänittäminen ei ollut mahdollista. Kysymykset lähetettiin haastateltaville etukäteen, jotta heillä olisi mahdollisuus tutustua aihepiiriin etukäteen. Koska kysymykset liittyivät tiiviisti viitekehykseen, kysymysten lähettämisellä etukäteen ei oletettavasti ole vaikutusta saatuihin vastauksiin, sillä haastateltavat eivät voineet valmistella vastauksiaan etukäteen.

Saunders et al. (2009, s. 490) mukaan laadullisen aineiston analysointiin ei ole standardoitua tapaa, mutta analysointi voidaan jakaa kolmeen vaiheeseen: ensin aineistosta tehdään yhteenveto, toiseksi aineisto kategorisoidaan ja kolmanneksi aineisto jäsennetään. Yhteenvetona, haastatteluiden vastaukset kerättiin taulukkoon, johon ne jaoteltiin

vastaajan ja kysymyksen mukaan. Vastaukset litteroitiin äänitteiden pohjalta sanasta sanaan, jotta vastaukset saatiin taltioitua mahdollisimman totuudenmukaisesti. Seuravaksi vastaukset ryhmiteltiin niistä tunnistettuihin kategorioihin: nykytila kohdeyksikön projekteissa, viitekehys ja käyttäjien osallistaminen yleisesti, iteraatio 0, käytettävyyspalaveri, käyttäjät katselmoinnissa ja tiimijaottelu. Myös seuraavassa luvussa tulokset on jäsennelty näiden kategorioiden mukaan, sekä tiivistetty taulukoihin 2-6. Lisäksi aineistoa jaoteltiin sen mukaan, sisälsivätkö ne omaan kategoriaansa liittyen haasteita, hyötyjä tai kehitysehdotuksia.

5. TULOKSET

5.1 Nykytila kohdeyksikön projekteissa

Haastatteluissa nousi esiin, että kohdeyksikön projekteissa otetaan käyttäjät huomioon hyvin eri tavoin asiakkaasta, projektitiimistä ja työtehtävistä riippuen.

”Ei ole strukturoitua tapaa (ottaa käyttäjiä tai käytettävyyttä huomioon), vaan kukin konsultti tekee työtään vähän omalla tavallaan.” (Projektipäällikkö, H2)

Yleisin käytäntö on, että käyttäjiin kiinnitetään huomiota pohtimalla itse tai projektitiimin kesken, millaisia käyttäjien käyttötapaukset ovat, miten ohjelmistosta saisi mahdollisimman yksinkertaisesti käytettävän ja miten käyttäjän toimintoja olisi mahdollista helpottaa. Tuoteasiantuntijat ja projektipäälliköt saattavat olla käyttäjien kanssa tekemisissä ohjelmiston valmistuttua, kouluttaessaan heille kuinka ohjelmistoa käytetään, mutta teknisemmillä henkilöillä, kuten ohjelmistokehittäjillä ja IT-arkkitehdeilla, ei ole suoraan käyttäjiin liittyviä vastuutehtäviä. Yleensä käyttäjien tarpeet välittyvät ohjelmistokehittäjille joko tuoteasiantuntijan tai asiakkaan kautta. Osa haastateltavista mainitsi, että usein asiakkaan projektitiimiin sisältyy pääkäyttäjä tai muu vastaava ”käyttäjien tulkki”, joka osallistuu ohjelmiston suunnitteluun ja antaa siitä säännöllisesti palautetta. Näissä haastatteluissa loppukäyttäjä määriteltiin henkilöksi, joka ei kuulu tiiviisti ohjelmiston kehityksessä mukana olevaan asiakkaan projektitiimiin, sillä tiiviisti projektissa olevat henkilöt eivät enää vastaa ohjelmiston osalta taitotasoltaan keskimääräistä loppukäyttäjää. Sen sijaan kaikki asiakkaan projektitiimissä olevat henkilöt lukeutuvat asiakkaaksi.

Haastatteluiden aikana mainituista projekteista kolmessa tehdään käyttäjien kanssa enemmän yhteistyötä: Näistä ensimmäisessä järjestetään iteraatioiden lopuksi demoja, joissa esitellään iteraation aikana valmistunut tuoteversio. Demoihin kutsutaan myös loppukäyttäjiä antamaan tuoteversiosta palautetta. Demot vastaavat pitkälti viitekehyksen katselmointeja, ja myöhemmin demojen ja katselmoinnin välille ei tehdä eroa, vaan molemmista käytetään käsitettä katselmointi. Lisäksi käyttäjiltä kysytään tarpeen mukaan palautetta ja ideoita iteraatioiden aikana. Tässä projektissa toimitaan hyvin samalla tavalla kuin viitekehyksessä suositellaan:

”Toi prototyyppi on uus juttu, muuten tää (viitekehys) vaikuttaa aika samalta mitä nyt tehdään. Ottaa tää (viitekehys) enemmän käyttäjää huomioon, esim. käyttäjien palautteen pohjalta muutokset suunnitelmiin.” (Projektipäällikkö, H3)

Toisessa projektissa pidetään ennen ensimmäistäkään iteraatiota käytettävyyspalaveria vastaavia työpajoja, joissa selvitetään haastatteluiden avulla, mitkä käyttäjien tarpeet ovat, mitä työtehtäviä heillä on ja millaiset ovat heidän työtapansa. Tarkoituksena on ymmärtää paremmin, miten järjestelmä palvelee käyttäjiä mahdollisimman hyvin, jo ennen toiminnallisuuksien toteuttamista. Kuitenkin asiakkaan projektitiimi päättää mitä järjestelmään tehdään ja miten se tehdään, joten heilläkin olisi hyvä olla näkemys käyttäjien todellisista tarpeista.

Kolmannessa projektissa käyttäjien kanssa ollaan tiiviimmässä yhteistyössä vasta käyttöönoton jälkeen ja käyttäjien kanssa käydään yhdessä läpi kuukausittain, mitä muutoksia he toivoisivat käytössä olevaan järjestelmään. Kaikissa kolmessa projektissa on koettu, että yhteistyö käyttäjien on kannattavaa.

Kaikki kohdeyksikön projektit ovat kuitenkin hyvin erilaisia. Budjetin, aikataulun, toteuttavan projektitiimin ja asiakkaan lisäksi projektit eroavat toisistaan projektin lähtökohtien osalta: Osa asiakkaista tietää hyvin tarkkaan jo projektin alussa, mitä he haluavat järjestelmältä. Toiset asiakkaista tarvitsevat toimittajalta enemmän tukea tarpeiden ja ratkaisujen selvittämisessä. Projektitiimin on helpompi selvittää tarpeita ja ratkaisuita, mikäli he ymmärtävät asiakkaan liiketoimintaa ja jokapäiväisiä työtehtäviä.

Kun haastateltavilta kysyttiin, mitä ongelmia viitekehyksen käyttöönotto voisi ratkaista, ratkaistavat ongelmat liittyivät enimmäkseen käytettävyysongelmien huomaamiseen liian myöhään. Lisäksi keskustelujen ja tekstien varaan nojaava kommunikointi aiheuttaa riskin, että eri osapuolet ymmärtävät suunnitelmat eri tavalla. Jos käyttäjät näkevät ohjelmiston vasta, kun se on valmis, he pystyvät kertomaan vasta valmiista ohjelmistosta, mikäli se ei vastaa heidän tarpeitaan. Jos jokin ominaisuus ei palvele käyttötarkoitustaan, niin sen eteen tehty työ menee hukkaan ja joudutaan tekemään asioita uudestaan. Pahimmassa tapauksessa käytettävyys voi jopa huonontua edelliseen käyttäjien käyttämään järjestelmään verrattuna.

”Tähän mennessä ei oikein ole käytettävyyttä edes mietitty, eli jos asiakas jotain haluaa, niin se vain toteutetaan.” (Sovelluskehittäjä, H7)

Toisinaan käykin niin, että käyttäjät käyttävät vain työtehtäviensä kannalta pakollista osuutta järjestelmästä ja muut kalliillakin rakennetut ominaisuudet jäävät pienen ryhmän käyttöön. Kaikki haastatteluun osallistuneet olivatkin yhtä mieltä siitä, että käyttäjiä ja käytettävyyttä pitäisi ottaa projekteissa nykyistä paremmin huomioon. Yksi syy sille on, että suurin osa asiakkaista pitää tärkeänä, että järjestelmä palvelee kaikkia käyttäjiä.

5.2 Viitekehys kohdeyksikön projekteissa

Tässä alaluvussa käsitellään ensin, millaisia ajatuksia haastateltavilla nousi viitekehyksestä yleisellä tasolla. Seuraavissa alaluvuissa 5.2.1-5.2.4 käsitellään tarkemmin viitekehyksen yksittäisiin osa-alueisiin, iteraatioon 0, käytettävyyspalaveriin, katselmointiin ja tiimijaotteluun, liittyneet mielipiteet haasteineen ja kehitysehdotuksineen. Taulukoihin 4-15 on kerätty jokaisesta alaluvusta keskeisimmät tulokset. Taulukot sisältävät eri osa-alueiden hyödyt ja hyötyjen kuvaukset, tai haasteet, haasteiden kuvaukset ja kehitysehdotukset. Yleisemmin viitekehykseen liittyvät hyödyt on tiivistetty taulukkoon 2.

Taulukko 2. *Käyttäjien osallistamisen hyödyt*

Hyödyt	**Kuvaus**
Kehitys aitojen tarpeiden pohjalta	*Tällä hetkellä yrityksessä kehitetään toiminnallisuuksia arvausten pohjalta. Käyttäjien osallistaminen mahdollistaa toiminnallisuuksien kehittämisen käyttäjiltä kerätyn tiedon ja palautteen pohjalta.*
Käytettävyysongelmien huomaaminen ajoissa	*Kun käyttäjille annetaan mahdollisuus antaa palautetta toiminnallisuuksista jo ennen käyttöönottoa, käytettävyysongelmat on mahdollista huomata aiemmin.*
Käyttäjien sitoutuminen ja luottamus toimittajaan	*Käyttäjiä osallistamalla ja heidän palautettaan kuuntelemalla voidaan osoittaa käyttäjille jo ennen käyttöönottoa, että järjestelmää tehdään heitä varten. Käyttäjät on mahdollista sitouttaa järjestelmän käyttöön paremmin, kun järjestelmään toteutetaan heidän tarpeidensa mukaisia toiminnallisuuksia.*

Viitekehyksessä pidettiin erityisesti siitä, että käyttäjiä osallistetaan projektin toteutukseen, alusta asti:

”Kun me suunnitellaan järjestelmiä, niin me ollaan kuitenki käyttäjää tottuneempia käyttämään järjestelmiä, niin sille sokeutuu millasta on käyttää järjestelmää aluksi. Käyttäjien mukaan ottaminen helpottais näkemään että mikä on vaikea hahmottaa harjaantumattomalle silmälle.” (Projektipäällikkö, H2)

Kun projektin aikana tiedetään käyttäjien todelliset tarpeet, projekti on mahdollista toteuttaa tarpeiden mukaisesti alusta lähtien, tai ainakin ongelmat huomattaisiin ajoissa ja mahdolliset käyttöönottoa seuraavat ikävät yllätykset jopa huonontuneen käytettävyyden osalta voitaisiin välttää. Lisäksi suoraan käyttäjältä kysyminen säästää aikaa, kun asiakkaan ei tarvitse toimia välikätenä. Käyttäjien osallistaminen voi myös lisätä käyttäjien sitoutuneisuutta järjestelmän käyttöön sen valmistuttua, ja lisätä luottamusta käyttäjien ja toimittajan välillä jo projektin alussa.

Jos asiakkaan kanssa joudutaan tilanteeseen, jossa useita toiminnallisuuksia joudutaan tekemään uudelleen, niin pelkästään jo keskustelut budjetista ja aikataulusta ovat kalliita, sillä ne vievät hyvin paljon aikaa molemmilta osapuolilta. Sen lisäksi asiakas ei välttämättä suostu maksamaan uudelleen tehtävästä työstä, mikä on hyvin kallista toimittajalle. Jos viitekehyksen avulla saadaan parannettua projektien onnistumista, niin se kannattaa ottaa käyttöön.

Viitekehystä ei kuitenkaan pidetty täysin ongelmattomana, kuten taulukkoon 3 kerätyistä haasteista ja kehitysehdotuksista on nähtävissä.

Taulukko 3. *Käyttäjien osallistamisen haasteet ja kehitysehdotukset*

Haasteet	**Kuvaus**	**Kehitysehdotukset**
Työmäärän kasvu	*Jos projektin tapahtumissa on paljon osallistujia, projektitiimi joutuu tekemään enemmän töitä mm. kerätyn tiedon jäsentelyssä.*	Työmäärän kasvun huomioiminen myyntivaiheessa ja työajan resursoinnissa.
Aikataululliset haasteet	*Mitä enemmän osallistujia projektin tapahtumissa on, sitä kauemmin tapahtumiin kuluu aikaa.*	Käyttäjien osallistamiseen kuluvan ajan huomioiminen jo myyntivaiheessa, sekä aikarajojen asettaminen projektin tapahtumiin.
Lisäkustannukset	*Asiakas voi olla haluton maksamaan ylimääräisiä kustannuksia käyttäjien osallistamisesta*	Viitekehyksestä voidaan tarjota asiakkaalle useita versioita, joista osa on kevyempiä kuin toiset.

Haasteet	Kuvaus	Kehitysehdotukset
Oikeiden käyttäjien löytäminen	*Eri toiminnallisuudet ovat hyödyllisiä eri käyttäjille, jolloin samat käyttäjät eivät voi arvioida kaikkia toiminnallisuuksia.*	Projektin alussa kannattaa määritellä, mitä käyttäjäryhmiä asiakkaalla on. Asiakkaalle voidaan antaa vastuu käyttäjien valinnasta ja oikeiden käyttäjien löytämisestä.
Kokonaisuuden hallinnan vaikeutuminen	*Kokonaisuuden hallinta (mm. aikatauluttaminen ja työnjako) vaikeutuu, kun projektissa on enemmän osallistujia.*	Asiakkaiden ja käyttäjien välillä tulee olla selkeä vastuunjako: mitä päättää asiakas, ja mitä kysytään käyttäjältä.

Viitekehyksen käyttöönotosta seuraisi työmäärän kasvua ja aikataulullisia haasteita, kun käytettävyydestä keskusteluun kuluisi enemmän aikaa. Sekä ajankäytön että työmäärän lisääntyminen kasvattaisi myös asiakkaan laskua, joten asiakas pitäisi saada vakuutettua siitä, että projektin toteuttaminen käyttäjäkeskeisesti on lisälaskun arvoista. Koska kaikki asiakkaat eivät ole valmiita maksamaan käyttäjäkeskeisyydestä täyttä hintaa, viitekehyksestä voisi olla eri laajuisia ja eri hintaisia. Kun projekti myydään asiakkaalle käyttäjäkeskeisenä, niin käyttäjäkeskeisyys pitää merkitä selvästi suunnitelmiin jo myyntivaiheessa, jotta viitekehys ei jää noudattamatta kiireen takia, jotta lisääntynyt työmäärä huomioidaan varmasti budjetissa ja työmääräarviossa, ja jotta myös asiakas sitoutuu laittamaan aikaansa projektiin alusta alkaen. Lisäksi käyttäjien ottaminen mukaan projektiin hankaloittaisi kokonaisuuden hallintaa entisestään, ja käyttäjien paikalle saaminen voi olla haasteellista:

”Kun halutaan osallistaa käyttäjiä, niin heidän käyttämä käyttämä työaika pitää laskea mukaan ja heille pitää lähettää kalenterikutsut ja kaikki hyvissä ajoin, että he pystyvät järjestelemään työnsä niin että he pääsevät osallistumaan.” (Projektipäällikkö, H5)

Osa haastateltavista koki, että voi olla vaikeaa löytää sellaisia käyttäjiä, jotka osaavat antaa palautetta ja kehitysehdotuksia iteraation aikana tehtyihin toiminnallisuuksiin. Haastatteluissa nousikin esiin, että erilaiset käyttäjäryhmät pitää määritellä projektin alussa ja eri käyttäjäryhmiä pitäisi osallistaa vain heitä koskevien toiminnallisuuksien

suunnitteluun ja arviointiin. Kaikkia käyttäjäryhmiä ei oteta tällä hetkellä huomioon, joten tunnistamalla eri käyttäjäryhmät, käyttäjillä on paremmat mahdollisuudet tulla kuulluksi projektin aikana. Toiminnallisuuksien kannalta oleellisten käyttäjäryhmien edustajien valitseminen voisi olla asiakkaan vastuulla. Lisäksi käyttäjien ja asiakkaan roolit pitää tehdä selväksi: käyttäjän vastuulla on kertoa tarpeet, ja asiakkaiden vastuulla on prosessien määrittely ja päätöksenteko. Usein käytettävyydestä joudutaankin joustamaan, kun prosessien pitää toimia täsmälleen oikein. Lisäksi, jos käytettävyyteen keskitytään liikaa, järjestelmä voi muuttua vaikeasti ylläpidettäväksi.

Haastatteluissa mainittiin useasti, että viitekehyksen toimivuuden käytännössä ja todelliset kehityskohteet on mahdollista havaita vasta, kun viitekehystä on kokeiltu käytännössä. Käytännönpuute onkin hyvä huomioida tämän tutkimuksen rajoitteena.

5.2.1 Iteraatio 0 ja prototyypin hyödyntäminen

Iteraation 0 tärkeimpinä osa-alueina pidettiin erilaisten käyttötapojen ja käyttäjäryhmien selvittämistä, ydintoimintojen suunnittelua käyttäjien aitojen tarpeiden pohjalta ja käyttäjien osallistaminen projektin alusta asti, kuten taulukosta 4 on nähtävissä.

Taulukko 4. *Iteraatioon 0 liittyvät hyödyt*

Hyödyt	**Kuvaus**
Kehitys aitojen tarpeiden pohjalta	*Vain käyttäjät tietävät todelliset tarpeensa, joten käyttäjiä osallistamalla voidaan saada selville, millaisia toiminnallisuuksia kannattaa kehittää.*
Käyttäjäryhmien kartoittaminen	*Kartoittamalla projektin kannalta oleelliset käyttäjäryhmät jo projektin alkuvaiheessa mahdollistaa oikeiden käyttäjien löytämisen eri toiminnallisuuksien suunnitteluun ja arviointiin.*
Käyttäjät mukana alusta asti	*Projektissa lähdetään jo alussa oikeaan suuntaan, kun käyttäjät arvioivat suunnitelmat ennen iteraatioita.*

Iteraation 0 avulla projektissa lähdetään alusta asti kehittämään oikeita asioita ja koska vain käyttäjät voivat tietää todelliset tarpeensa, kehitystä lähdettäisiin tekemään oikeiden tarpeiden pohjalta:

”Kaikista tärkeintä on se, että se pohjatyö on tehty hyvin. Että meillä on oikeesti

hyvät käyttötapaukset, joiden avulla oikeesti ymmärretään se tarve, mikä asiakkaalla on ja missä ja miten ne toimii. Jos me ymmärretään vääriin niin aletaan meneen mönkään ihan kokonaan." (Projektipäällikkö, H5)

Iteraation 0 osalta haastatteluissa nousi esiin myös kehitysehdotuksia ja haasteita, joita pitäisi ottaa huomioon iteraation 0 aikana. Ne on tiivistetty taulukkoon 5.

Taulukko 5. *Iteraatioon 0 liittyvät haasteet ja kehitysehdotukset*

Haasteet	**Kuvaus**	**Kehitysehdotukset**
Käyttäjät haluavat uuden järjestelmän olevan samanlainen kuin edellinen	*Käyttäjät voivat odottaa, että uusi järjestelmä olisi samankaltainen kuin edellinen ja tyrmätä suunnitelmat, jotka poikkeavat vanhasta järjestelmästä.*	Järjestelmä voidaan esitellä käyttäjille jo ennen iteraatiota 0, jotta käyttäjät näkevät, että järjestelmä on erilainen kuin edellinen.
Valmiin järjestelmän asettamat rajoitteet	*Jos projektissa kustomoidaan valmista järjestelmää, kaikkien toiveiden täyttäminen ei ole mahdollista. Käyttäjille voi olla vaikeaa ymmärtää, että kaikkea ei voida toteuttaa.*	Projektitiimillä tulee olla ymmärrys järjestelmän asettamista rajoitteista. Järjestelmä voidaan esitellä käyttäjille jo ennen iteraatiota 0, jotta myös käyttäjät voivat hahmottaa järjestelmän rajoitteita.
Työmäärän lisääntyminen	*Iteraatio 0 on ylimääräinen vaihe ennen muita iteraatioita, joten se lisää yhden iteraation verran työmäärää.*	Asiakkaalle voidaan antaa vastuu analyysivaiheesta, jolloin asiakas toteuttaa itse käyttäjäryhmien määrittelyn ja käyttötapausten kartoittamisen.

Haasteet	Kuvaus	Kehitysehdotukset
Etukäteen tehdyt suunnitelmat voivat rajoittaa projektin ketteryyttä	*Ketterässä ohjelmistokehityksessä pyritään suunnittelemaan mahdollisimman vähän etukäteen, jotta muutoksiin reagoiminen on helppoa ja nopeaa.*	Iteraation 0 aikana keskitytään vain käyttäjän tarpeiden kartoittamiseen ja toiminnallisuuksiin, joilla tärkeimmät tarpeet voidaan täyttää.

Käyttäjät haluavat usein järjestelmän olevan vain parempi versio heidän vanhasta järjestelmästään, mikä ei ole paras lähtökohta ohjelmiston kehitykselle. Lisäksi kohdeyksikön projekteissa kustomoidaan valmiita tuotteita, mikä asettaa toteutukselle rajoitteet. Jos järjestelmä näytettäisiin käyttäjille jo ennen iteraatiota 0, käyttäjät hahmottaisivat paremmin, millaisia järjestelmä ja sen käyttöliittymä ovat, etenkin rajoitusten osalta. Iteraatioon 0 osallistuvien projektitiimiläistenkin on ymmärrettävä järjestelmän rajoitukset, jotta suunnitelmat on mahdollista toteuttaa myöhemmissä iteraatioissa.

Ensimmäisessä kehitysehdotuksessa kevennettäisiin iteraatiota 0 antamalla asiakkaalle vastuun analyysivaiheesta. Asiakas voisi itsenäisesti kartoittaa käyttäjien toiveet ja tarpeet, jonka jälkeen projektitiimi ottaa vastuun lopuista iteraation 0 vaiheista. Analyysivaihe voisi sujua nopeammin asiakkaan toteuttamana ja projektitiimi säästäisi muutenkin aikaa, kun heidän ei tarvitsisi osallistua analyysivaiheeseen. Projektin toteuttaminen helpottuu, jos asiakas tietää käyttäjien todelliset tarpeet.

Riskinä iteraatiossa 0 nähtiin se, että suunnitelmien määrittelyssä mentäisiin liian pitkälle jo projektin alkuvaiheessa. Usein vasta projektin edetessä selviää, mitä oikeasti kannattaisi tehdä, ja liian tarkat suunnitelmat rajoittavat projektin ketteryyttä. Tarkan suunnitelman sijaan iteraatiossa 0 tulisi keskittyä käyttäjien tarpeisiin ja toiminnallisuuksiin, joilla tarpeet voidaan täyttää:

” Enemmänkin voidaan vain keskustella asioista yleisellä tasolla: nämä lomakkeet voisi toteuttaa netissä, kun ne ovat aiemmin tehty paperilla. Ei mennä vielä sisältöön, vaan tehdään isompi kuva." (IT-arkkitehti, H8)

Käyttäjät arvioisivat suunnitelmat karkealla tasolla ja kertoisivat, jos jotakin ei ole vielä otettu huomioon. Käyttäjien palautteen pohjalta suunnitelmia voidaan tarvittaessa muuttaa.

Iteraatiossa 0 ja käytettävyyspalaverissa hyödynnettävä prototyyppi herätti paljon ajatuksia puolesta ja vastaan. Prototyyppiin liittyvät haasteet ja kehitysehdotukset on koottu taulukkoon 6 ja siihen liittyvät hyödyt taulukkoon 7.

Taulukko 6. *Prototyyppiin liittyvät haasteet ja kehitysehdotukset*

Haasteet	**Kuvaus**	**Kehitysehdotukset**
Iteraatio 0 on liian varhainen vaihe prototyypille	*Iteraatiossa 0 ei kannata suunnitella liikaa, jotta suunnitelmat eivät rajoita myöhempää kehitystä. Prototyypin hyödyntäminen tekee suunnitelmista konkreettisia, joka voi itsessään rajoittaa myöhempää kehitystä.*	Prototyyppejä kannattaa tehdä vasta käytettävyyspalaverissa
Ajankäytön lisääntyminen	*Prototyypin tekeminen vie aikaa varsinkin, kun iteraatiossa kehitetään useita toiminnallisuuksia.*	Prototyyppejä tulee tehdä vain tärkeimpien ominaisuuksien osalta ja pienet kustomoinnit voidaan tehdä suoraan järjestelmään

Kaikki haastatteluun osallistuneet pitivät prototyyppiä hyödyllisenä työkaluna käytettävyyspalaverissa, mutta osa vastanneista koki iteraation 0 liian varhaiseksi vaiheeksi prototyypin hyödyntämiselle:

”En tiedä päästäänkö ikinä niin pitkälle, että analyysivaiheen lopputuloksena olis prototyyppi, mutta vaikka olis vaan speksit siinä vaiheessa niin niistäkin olis apua. Sitten kyllä pitäisi olla prototyyppi viimeistään sprinteissä, että voisi näyttää käyttäjille missä mennään.” (Ohjelmistokehittäjä, H9)

Lisäksi prototyypin tekeminen vie aikaa, eikä kaikista toiminnallisuuksista ja ohjelmiston osa-alueista ole mahdollista tehdä prototyyppiä. Siitä syystä ohjelmistosta pitäisi valita tärkeimmät ominaisuudet prototyyppiin. Toisinaan prototyypin tekeminen ei myöskään ole tarpeellista, sillä osa toiminnallisuuksista voidaan samassa ajassa kustomoida suoraan järjestelmään.

Taulukko 7. *Prototyyppiin liittyvät hyödyt*

Hyödyt	Kuvaus
Suunnitelmien konkreettisuus	*Suunnitelmien välittäminen asiakkaalle ja muille sidosryhmille on helpompaa konkreettisessa muodossa, ja väärinkäsitysten riski pienenee*
Käyttäjien ja asiakkaiden luottamuksen vahvistuminen suunnitelmia kohtaan	*Suunnitelmien näkeminen konkreettisessa muodossa voi lisätä käyttäjien ja asiakkaiden luottamusta projektia kohtaan jo suunnitteluvaiheessa, kun he voivat nähdä järjestelmän jo ennen sen valmistumista*
Toiminnallisuuksien kehittäminen helpottuu	*Myös projektitiimin ohjelmistokehittäjät hyötyvät prototyypeistä: konkreettisista suunnitelmista on mahdollista nähdä toivottu lopputulos ja kehittää sen mukaisia ratkaisuja*

Prototyypistä pidettiin erityisesti siksi, että sen avulla suunnitelmat voitaisiin näyttää konkreettisessa muodossa käyttäjille ja asiakkaalle:

”Parasta ja hyödyllisintä feedbackiä saa asiakkailta ja käyttäjiltä, kun on jotain kättä pidempää mitä näyttää, koska muuten niistä puhutaan vain puheen tasolla, jolloin on riski, että eri osapuolet ymmärtää asian eri tavalla. Sen takia prototyyppi olisi hyödyllinen kommunikoinnin välineenä.” (Projektipäällikkö, H2)

Suunnitelmien esittäminen käyttäjille prototyypin muodossa näyttäisi käyttäjille, että heitä on kuunneltu, ja se vahvistaisi luottamusta projektitiimin, käyttäjien ja asiakkaan välillä alusta lähtien. Lisäksi prototyyppi helpottaa projektitiimin sisäistä tiedon jakoa, sillä konkreettisten suunnitelmien pohjalta on helpompi lähteä kehittämään suunnitelmien mukaista ohjelmistoa.

5.2.2 Käytettävyyspalaveri

Käytettävyyspalaveri jakoi jonkin verran haastatteluun osallistuneiden mielipiteitä ja siihen kohdistui työmäärään ja käytettävyyspalaverin ajankohtaan liittyviä haasteita ja kehitysehdotuksia, jotka ovat nähtävissä taulukosta 8.

Taulukko 8. *Käytettävyyspalaveriin liittyvät haasteet ja kehitysehdotukset*

Haasteet	**Kuvaus**	**Kehitysehdotukset**
Työmäärän ja ajankäytön lisääntyminen iteraatioissa	*Käytettävyyspalaveri vie aikaa ja lisää projektitiimin työtä varsinkin, jos käyttäjiltä kysytään palautetta kaikista toiminnallisuuksista ja jos heidän tarpeitaan kartoitetaan useissa iteraatioissa.*	Keskitytään vain tärkeimpiin toiminnallisuuksiin. Kevyemmässä versiossa käyttäjiltä kysytään palautetta suunnitelmista, jotka projektitiimi tekee täysin itsenäisesti, kartoittamatta ensin käyttäjien tarpeita toiminnallisuuksiin liittyen.
Käytettävyyspalaverin järjestäminen heti suunnittelupalaverin jälkeen	*Kaikissa projekteissa ei saada täyttä hyötyä käytettävyyspalaverista, jos se järjestetään heti suunnittelupalaverin jälkeen. Osa toiminnallisuuksista on parempi suunnitella vasta, kun projektitiimillä on parempi käsitys niiden toteuttamismahdollisuuksista.*	Käytettävyyspalaveri voidaan järjestää joustavasti: iteraatiosta riippuen, se voi olla heti suunnittelupalaverin jälkeen tai joidenkin toiminnallisuuksien kehittäminen voidaan aloittaa jo ennen käytettävyyspalaveria.

Käytettävyyspalaverin koettiin olevan liian raskas, eli lisäävän työmäärää liian paljon verrattuna käytettävyyspalaveriin saatuihin hyötyihin ja vievän paljon aikaa iteraatiosta. Käytettävyyspalaveriin liitty myös riski, että sen aikana keskitytään vääriin asioihin, etenkin jos siihen osallistuu useita käyttäjiä. Käytettävyyspalaverille toivottiin kevyempää versiota, ja selkeämpää toiminnallisuuksien määrittelyvaihetta jo ennen käytettävyyspalaveria. Lisäksi ennen käytettävyyspalaveria pitäisi selvittää, mitkä toiminnallisuudet ovat oleellisimpia käyttäjien työtehtävien kannalta, jotta käytettävyyspalaverissa voitaisiin keskittyä tärkeimpiin toiminnallisuuksiin.

Käytettävyyspalaverissa nähtiin myös paljon hyviä puolia (taulukko 9). Suunnitelmien läpikäyminen uudelleen sprintin aikana pidettiin hyödyllisenä, sillä ymmärrys tarpeista ja

lopputuotteesta paranee käyttäjillä, asiakkaalla ja projektitiimilläkin projektin edetessä. Suunnittelupalaverin ja käytettävyyspalaverin pitämistä täysin erillään toisistaan nähtiin hyväksi asiaksi, koska käyttäjistä ei uskottu olevan hyötyä vielä suunnittelupalaverissa. Lisäksi käytettävyyspalaverin koettiin ehkäisevän turhaa työtä:

”Toteutus ei menisi hukkaan, kun loppukäyttäjän kanssa olisi etukäteen katsottu, että tällaiset toiminnallisuudet täyttäisivät tällaiset tarpeet. Että onko se hyvä vai muokataanko sitä heti siinä kohtaa.” (IT-arkkitehti, H8)

Taulukko 9. *Käytettävyyspalaveriin liittyvät hyödyt*

Hyödyt	**Kuvaus**
Suunnitelmat käydään läpi uudelleen käyttäjien kanssa	*Ketterälle ohjelmistokehitykselle on ominaista, että suunnitelmat muuttuvat, jonka vuoksi iteraatiossa 0 tehtyjen suunnitelmien läpi käyminen uudelleen käyttäjien kanssa myöhemmissä iteraatioissa on hyödyllistä. Lisäksi projektitiimin tuntemus asiakkaan ja käyttäjien tarpeista kasvaa projektin edetessä, mikä voi vaikuttaa suunnitelmiin.*
Turhan työn ehkäiseminen	*Kun käyttäjät antavat palautetta suunnitelmista iteraatioiden alussa, voidaan kehittää alusta asti oikeanlaisia toiminnallisuuksia, jotka vastaavat käyttäjien tarpeisiin. Jos käyttäjät antavat palautetta vasta valmiista toiminnallisuudesta, projektitiimi on jo toteuttanut myös ne ongelmat, jotka olisi voitu havaita suunnitteluvaiheessa.*

Konkreettisimmat kehitysehdotukset liittyivät käytettävyyspalaverin keventämiseen ja joustavuuteen. Iteraatioihin tulisi lisää joustavuutta, jos käytettävyyspalaveri voidaan pitää joissain iteraatioissa suunnittelupalaverin jälkeen, kuten viitekehyksessä ehdotetaan, mutta monimutkaisemmissa iteraatioissa toiminnallisuuksien kehittäminen voitaisiin aloittaa jo ennen käytettävyyspalaveria, ja jatkaa kehitystä edelleen käytettävyyspalaverin jälkeen. Kun kehitystä tehtäisiin jo vähän etukäteen, projektitiimillä voisi olla parempi tietämys siitä, mitä käytettävyyspalaverissa kannattaa käydä läpi ja mihin asioihin tarvitaan käyttäjien palautetta.

Käytettävyyspalaverille ehdotettiin myös kevyempää mallia, jossa projektitiimi suunnittelisi iteraatioon valitut toiminnallisuudet aiemman kokemuksen pohjalta, kartoittamatta

käyttäjien tarpeita iteraation aikana. Valmiit suunnitelmat näytettäisiin käyttäjille prototyypin muodossa, ennen toiminnallisuuksien kehittämisen aloittamista. Käyttäjät voisivat arvioida karkealla tasolla, sopivatko suunnitelmat heidän työtehtäviinsä ja tarpeisiinsa, ja antaa muutosehdotuksia tarvittaessa. Kun käyttäjät ovat arvioineet suunnitelmat, tuoteversiota lähdettäisiin varmemmin kehittämään oikeaan suuntaan. Katselmoinnissa käyttäjät pääsisivät arvioimaan valmiit ominaisuudet tarkemmin, jonka jälkeen mahdolliset korjaukset voidaan vielä tehdä seuraavassa iteraatiossa.

5.2.3 Käyttäjät katselmoinnissa

Käyttäjien osallistuminen katselmointiin koettiin yleisesti hyväksi käytännöksi, ja katselmoinnille tunnistettiin useita hyötyjä, jotka on koottu taulukkoon 10.

Taulukko 10. *Katselmointiin liittyvät hyödyt*

Hyödyt	**Kuvaus**
Havaitut ongelmat voidaan korjata seuraavassa iteraatiossa	*Ongelmien korjaaminen on helpompaa ja kustannustehokkaampaa, mitä aiemmin ongelmat huomataan. Ongelmat eivät myöskään haittaa käyttäjiä, jos ne havaitaan ja korjataan ennen käyttöönottoa.*
Palaute useilta käyttäjäryhmiltä	*Kun palautetta saadaan käyttäjäryhmiltä, jotka tulevat käyttämään toteutettuja toiminnallisuuksia, voidaan selvittää mahdollisimman luotettavasti, täyttävätkö toiminnallisuudet heidän tarpeensa.*
Uudet, käyttäjien tarpeisiin pohjautuvat ideat	*Katselmointi tarjoaa käyttäjille mahdollisuuden kertoa toteutusideoistaan projektitiimille. Uusia ideoita voi syntyä, kun käyttäjät näkevät, mitä järjestelmän avulla voidaan tehdä.*
Käyttäjien sitouttaminen	*Käyttäjät ovat sitoutuneempia käyttämään järjestelmää käyttöönoton jälkeen, kun he pääsevät osallistumaan sen toteutukseen ja havaitsemaan ongelmia jo ennen käyttöönottoa. Käyttöönotto on muutenkin helpompi, kun käyttäjät ovat päässeet tutustumaan järjestelmään jo aiemmin.*

Osa haastateltavista piti käyttäjien osallistamista katselmointiin viitekehyksen tärkeimpänä osana ja piti mahdollisena parantaa projektien käyttäjäkeskeisyyttä pelkästään osallistamalla käyttäjiä katselmointeihin, sillä sekin jo mahdollistaisi käytettävyysongelmien korjaamisen seuraavassa sprintissä ja palautteen saamisen eri käyttäjäryhmiltä:

”Se, että olis useampi käyttäjärooli edustettuna katselmoinnissa, niin sekin olisi jo parannus entiseen.” (Integraatiopäällikkö, H6)

Toisaalta, jos iteraation aikana käyttäjiltä kerättään palautetta vain katselmoinnissa, niin iteraation aikana arvailtaisiin, mitä käyttäjät haluavat ja tarvitsevat, ja vasta katselmoinnissa saataisiin palautetta iteraation aikana tehdyistä valinnoista ja toteutuksesta. Katselmointiin voi liittyä ongelmia myös silloin, jos siihen osallistuu liian useita käyttäjiä, jonka vuoksi käyttäjiä pitäisi valita katselmointiin vain tietty määrä (taulukko 11). Eräässä käyttäjiä osallistavassa projektissa käyttäjien osallistumismäärää ei kuitenkaan ole rajoitettu, vaan kaikki halukkaat ovat saaneet osallistua katselmointeihin ja antaa niissä palautetta tuoteversiosta. Tämän projektin osalta, haastatteluissa ei noussut negatiivisia kokemuksia myöskään käyttäjien määrän suhteen.

Taulukko 11. *Katselmointiin liittyvät haasteet ja kehitysehdotukset*

Haasteet	**Kuvaus**	**Kehitysehdotukset**
Työmäärän ja ajankäytön lisääntyminen iteraation aikana	*Käyttäjien osallistaminen iteraatioissa useassa eri tapahtumassa kuluttaa aikaa ja lisää työmäärää.*	Käyttäjien kutsuminen vain katselmointiin iteraation aikana.
Käyttäjien runsas lukumäärä	*Jos käyttäjiä osallistuu katselmointeihin liian paljon, katselmointiin voi kulua hyvin paljon aikaa, eikä kaikkien ideoita tai kehitysehdotuksia voida toteuttaa.*	Osallistujien lukumäärän rajoittaminen.

Osassa kohdeyksikön projekteista käyttäjiä on jo aiemminkin kutsuttu katselmointeihin antamaan palautetta tuoteversioista, jolloin projektitiimi on saanut käyttäjiltä uusia ideoita, ja projektitiimi on pysynyt paremmin selvillä käyttäjien tarpeista projektin aikana. Vaikka ensimmäisissä iteraatioissa ei välttämättä valmistu sellaisia tuoteversioita, joista

käyttäjät voisivat antaa palautetta, katselmoinnit toimivat projektin alussa käyttäjien sitouttamisen ja muutosjohtamisen tukena. Katselmoinneissa voidaan näyttää käyttäjille, että projekti etenee ja että projektissa tehdään käyttäjille tärkeitä asioita. Etenkin viitekehyksen avulla, kun käyttäjiä ja heidän tarpeitaan ymmärretään paremmin, katselmoinneissa voidaan panostaa käyttäjille tärkeimpien osa-alueiden esittelyyn.

5.2.4 Tiimijaottelu

Kaikki haastatteluun osallistuneet kokivat mahdolliseksi, että projekteissa jatketaan vain yhden tiimin voimin, vaikka projektit toteutetaan käyttäjäkeskeisemmin. Lisäksi suurin osa oli sitä mieltä, että yksi tiimi on parempi vaihtoehto kuin kaksi tiimiä, joista toinen keskittyisi vain käyttäjiin ja käytettävyyteen ja toinen varsinaiseen toteutukseen. Yhden tiimin hyödyntämisen hyödyt on tiivistetty taulukkoon 12.

Taulukko 12. *Yhden tiimin hyödyt*

Hyödyt	**Kuvaus**
Tiedon jakaminen	*Tiedon jakaminen yhdessä tiimissä on helpompaa kuin kahdessa, sillä yhdessä tiimissä kommunikointi on tiiviimpää.*
Ketteryys	*Yksi tiimi on ketterämpi kuin kaksi tiimiä, sillä muutoksia ei tarvitse hyväksyttää kahdessa tiimissä ja kommunikointi on tiiviimpää.*
Osaamisen jakaminen	*Samassa tiimissä on paremmat mahdollisuudet jakaa osaamista ja oppia muilta tiimiläisiltä. Esimerkiksi käytettävyyttä osaava voi jakaa tietämystään muille tiimiläisille ja muut tiimiläiset voivat jakaa tietämystään järjestelmän toiminnasta ja rajoitteista.*
Kaikkien osa-alueiden huomioiminen käytettävyydessä	*Koska kommunikointi on tiiviimpää yhdessä tiimissä ja kaikki tarvittava osaaminen löytyy samasta tiimistä, on helpompaa huomioida kaikki osa-alueet myös käytettävyyden osalta. Käytettävyyden parantaminen voi tarkoittaa muutoksia jopa integraatiotasolla.*

Yhden tiimin paremmuutta perusteltiin usein kommunikaation kautta:

”Yksi tiimi on aina kommunikaation kannalta parempi. Kahden tiimin kanssa käy väkisinkin niin, että kaikki viestit ei mene perille.” (Integraatiopäällikkö, H6)

Lisäksi yhden tiimin koetaan olevan ketterämpi, luovan paremmat mahdollisuudet osaamisen jakamiselle tiimin jäsenten välillä ja sopivan paremmin valmiin järjestelmän mukauttamiseen. Ketterässä ohjelmistokehityksessä voidaan lisätä tai poistaa toiminnallisuuksia kesken projektin, jolloin muutosten arvioiminen kahdessa tiimissä hankaloittaa työtä. Lisäksi käytettävyyden huomioiminen voi tarkoittaa muutoksia järjestelmän monilla tasoilla, joten tarvittavan osaamisen on oltava samassa tiimissä, jotta kaikki osa-alueet osataan ottaa huomioon. Käytettävyyden huomioimista kaikilla osa-alueilla tukee myös käytettävyysosaamisen jalkauttaminen kaikille tiimiläisille, mikä onnistuu paremmin yhdessä tiimissä:

”Samassa tiimissä opittais paremmin toisiltaan, että käytettävyysammattilainen oppis käytännöstä ja muut sitten oppis sitä käytettävyyttä ja se menis enemmän selkärankaan.” (Tuoteasiantuntija, H4)

Kuten aiemmin on mainittu, kohdeyksikön projekteissa kustomoidaan valmiita Microsoft Dynamics 365 järjestelmiä, mikä asettaa rajoituksia käyttäjien toiveiden täyttämiselle ja käytettävyyden hiomiselle. Samassa tiimissä tulee siis olla ymmärrys siitä, miten järjestelmä toimii ja millainen käytettävyyden pitäisi olla. Useissa haastatteluissa painotettiin, että tietämys järjestelmän toiminnasta ja käytettävyydestä olisi hyvä olla samalla henkilöllä, sillä ulkopuolinen käytettävyyden ammattilainen voi takertua sellaisiin alustan ominaisuuksiin, johon projektitiimi ei pysty vaikuttamaan (taulukko 13).

”Yhdessä projektissa oli tällainen (ulkopuolinen käytettävyyden ammattilainen) ja sitten heidän ratkaisut oli tosi kaukaa haettuja, kun he ei ymmärtäny sitä järjestelmää, siis tuntenut teknisiä rajoitteita.” (Tuoteasiantuntija, H4)

Taulukko 13. *Käytettävyysosaamiseen liittyvät haasteet ja kehitysehdotukset*

Haasteet	**Kuvaus**	**Kehitysehdotukset**
Käytettävyysosaamisen puute	*Projektitiimiläisillä ei ole tällä hetkellä käytettävyysosaamista.*	Ulkopuolinen käytettävyyden ammattilainen voi kouluttaa käytettävyyttä projektitiimiläisille projektin aikana.

Haasteet	Kuvaus	Kehitysehdotukset
Käytettävyysratkaisut eivät aina ole toteutettavia	*Varsinkaan ulkopuolinen käytettävyyden ammattilainen ei tunne valmiin järjestelmän rajoituksia, jolloin käytettävyysratkaisut eivät ole mahdollisia toteuttaa.*	Tietämys käytettävyydestä ja järjestelmän rajoitteista tulee olla samalla henkilöllä tai vähintäänkin samassa tiimissä.

Tiimin kouluttamisessa ulkopuolista käytettävyyden ammattilaista voitaisiin hyödyntää, varsinkin aluksi, kun tiimiläisillä ei vielä ole tarpeeksi osaamista käytettävyydestä. Yksittäisten koulutusten sijaan, tiimiä voisi kouluttaa antamalla vinkkejä ja kommentteja käytettävyydestä pitkin projektia.

Taulukko 14. *Kahden tiimin hyödyt*

Hyödyt	Kuvaus
Työtaakan voi jakaa kahdelle tiimille	*Jos projektissa on suuret työmäärät, tai käytettävyys on asiakkaalle todella tärkeää, työtaakka voi olla parempi jakaa kahdelle erilliselle tiimille.*
Kahden näkökulman saaminen ratkaisuehdotuksiin	*Eri tiimeissä olevasta erilaisesta osaamisesta voi olla hyötyä toiminnallisuuksien sisäisessä arvioinnissa.*

Kahden tiimin koetaan sopivan paremmin suurempiin projekteihin, joissa on enemmän resursseja ja joissa työmäärän voi jakaa luontevasti kahden tiimin välillä (taulukko 14). Kahta tiimiä voidaan tarvita myös sellaisissa projekteissa, joissa käytettävyys on selkeänä prioriteettina. Lisäksi hyötynä voidaan nähdä se, että ratkaisuihin saataisiin kaksi erillistä näkökulmaa.

Taulukko 15. *Kahden tiimin haasteet ja kehitysehdotukset*

Haasteet	Kuvaus	Kehitysehdotukset
Tiedon jakaminen	*Tiedon jakaminen kahden tiimin välillä on haasteellisempaa kuin yhden tiimin sisällä.*	

Haasteet	Kuvaus	Kehitysehdotukset
Ketteryyden säilyttäminen	*Muutokset tulee hyväksyttää molemmilla tiimeillä, jolloin muutosten tekeminen ei ole yhtä ketterää kuin yhden tiimin sisällä.*	
Korkeat kustannukset	*Kahden tiimin sitouttaminen yhteen projektiin on kalliimpaa kuin yhden tiimin.*	Toisen tiimin hyödyntäminen vain suurissa projekteissa tai käytettävyyden ollessa hyvin tärkeää asiakkaalle.
Tiimien välisen yhteistyön aikatauluttaminen	*Molemmilla tiimeillä voi olla useita projekteja yhtä aikaa, jolloin yhteistyön aikatauluttaminen voi olla hankalaa.*	

Kahden tiimin haasteeksi nostetaan tiimien väliseen kommunikaatioon liittyvien haasteiden lisäksi korkeat kustannukset ja ajanpuute (taulukko 15). Kahden tiimin yhteistyössä ongelmia seuraisi siis siitä, että molemmat tiimit tekisivät todennäköisesti montaa projektia yhtä aikaa, jolloin yhteisen ajan löytäminen eri tiimien välillä on hyvin vaikeaa. Jos tiimeillä ei ole tarpeeksi aikaa esimerkiksi yhteisille päiväpalavereille, tiedon jakaminen tiimien välillä on entistäkin haastavampaa.

6. POHDINTA

6.1 Käyttäjäkeskeisyyden ja ketteryyden yhteensovittamisen periaatteet teoriassa ja käytännössä

Tämän tutkimuksen pohjalta aiemmin esitettyihin käyttäjäkeskeisyyden ja ketteryyden yhteensovittamisen periaatteisiin (Brhel et al. 2015) ei tullut muita muutoksia, kuin periaatteen kolme ”Käyttäjäkeskeinen suunnittelu ja ohjelmistokehitys toteutetaan erillisissä ja erivaiheisissa iteraatioissa, jotka kuitenkin nivoutuvat yhteen” korvaaminen periaatteella, joka mahdollistaa käyttäjäkeskeisen ja ketterän ohjelmistokehitysprojektin toteuttamisen yhdellä tiimillä. Alaluvussa 3.2.6. korvaavaksi periaatteeksi ehdotettu ”Iteraatioihin lisätään käyttäjäkeskeisyyttä järjestämäl lä käytettävyyspalavereita ja kutsumalla käyttäjät katselmointeihin” näyttää tämän tutkimuksen valossa sopivan yhden tiimin toteuttamiin projekteihin, sillä kaikki haastatellut henkilöt olivat yhtä mieltä siitä, että näiden periaatteiden pohjalta toteutettu viitekehys olisi mahdollista toteuttaa yhdellä tiimillä. Tämän tutkimuksen perusteella käyttäjäkeskeisyyden ja ketteryyden yhteensovittamisen periaatteet ovat siis seuraavat:

1. Projektin alussa on erillinen suunnitteluvaihe.
2. Tuote toteutetaan iteraatioiden ja tuoteversioiden pohjalta
3. Iteraatioihin lisätään käyttäjäkeskeisyyttä järjestämällä käytettävyyspalavereita ja kutsumalla käyttäjät katselmointeihin.
4. Sidosryhmiä osallistetaan tuotteen kehitykseen aktiivisesti.
5. Tuotokset kommunikoidaan sidosryhmille helposti ymmärrettävässä ja konkreettisessa muodossa.

Seuraavaksi käsitellään näiden periaatteiden kautta, miten käyttäjäkeskeisen ketterän ohjelmistokehityksen kirjallisuus ja tässä tutkimuksessa toteutetut haastattelut tukevat toisiaan ja millaisia ristiriitoja niiden välillä esiintyi.

6.1.1 Iteraatio 0

Ensimmäisen periaatteen mukaan projektin alussa tulee olla erillinen suunnitteluvaihe, iteraatio 0 (Brhel et al. 2015). Haastatteluissa nousi esiin huoli siitä, että liiallinen suunnittelu projektin alussa rajoittaa projektin ketteryyttä. Myös Da Silva et al. (2018) mainitsevat, että yleensä ketterässä ohjelmistosuunnittelussa pyritään välttämään liiallista

suunnittelua. Kuitenkin sekä haastatteluissa, että kirjallisuudessa (Ferreira et al. 2007b) tunnistetaan iteraation 0 hyödyt: sen avulla on mahdollista välttää huonojen suunnittelupäätösten ja toteutusten tekeminen, jolloin kallista uudelleen suunnittelua ja toteutusta ei tarvita. Haastatteluissa nousi esiin, että iteraatioiden aikana toiminnallisuudet toteutetaan projektitiimin arvausten pohjalta ja toteutusten toimivuus voi selvitä vasta, kun järjestelmä otetaan käyttöön. Haastatteluissa oltiin yhtämieltä kirjallisuuden (Ferreira et al. 2007b; Kuusinen & Väänänen-Vainio-Mattila 2012; Salah et al. 2014; Lárusdóttir et al. 2017; Da Silva et al. 2018) kanssa siitä, että käytettävyysongelmien huomaaminen liian myöhään aiheuttaa ongelmia: toiminnallisuuksien toteuttaminen uudelleen aiheuttaa lisäkustannuksia, ja käyttäjien pettyessä järjestelmään he eivät halua käyttää järjestelmää, jolloin osa järjestelmän toiminnallisuuksista voivat jäädä hyödyntämättä. Iteraation 0 toteuttaminen on tasapainoilua sen välillä, mikä on tarpeeksi suunnittelua ja mikä on liikaa. Kuitenkin iteraatioissa on mahdollista jatkaa suunnittelua käytettävyyspalaverien aikana, joten joidenkin projektien osalta voi riittää, jos iteraation 0 aikana keskitytään käyttäjien ymmärtämiseen, ja käytettävyyden suunnittelu jätetään vähemmälle.

Fox et al. (2008) ja Da Silva et al. (2012) ehdottavat, että iteraation 0 aikana tehdyt suunnitelmat dokumentoidaan esimerkiksi prototyyppien muodossa. Osa haastateltavista piti kuitenkin iteraatiota 0 liian varhaisena vaiheena tehdä prototyyppejä, koska suunnitelmien tulee olla hyvin karkeita ennen varsinaisen toteutusvaiheen iteraatioita, jotta suunnitelmia ja toteutusta ei päätetä liian aikaisin, miikä rajoittaisi projektin ketteryyttä. Kuitenkin osa haastateltavista piti prototyyppien tekemistä hyvin tärkeänä, joten niiden hyödyntämistä iteraatiossa 0 tulee harkita projektikohtaisesti.

Da Silva et al. (2012) suosittelee ammattilaisten käytettävyyden suunnittelijoiden hyödyntämistä iteraatiossa 0, mutta haastatteluiden perusteella ainakin valmiin ohjelmiston suunnittelussa on tärkeämpää, että suunnittelijoilla on ymmärrys ohjelmiston asettamista rajoitteista. Haastattelussa viitattiin aiempiin kokemuksiin ulkopuolisen käytettävyyden ammattilaisen hyödyntämisestä, jolloin suunnitelmia ei ole ollut mahdollista toteuttaa, kun suunnittelijalla ei ole ollut tietämystä ohjelmistosta. Käytettävyyden ammattilaista voitaisiin kuitenkin hyödyntää, kuten haastatteluissakin nousi esiin, projektitiimiläisten kouluttamisessa ja he voisivat antaa projektitiimiläisille vinkkejä käytettävyyden huomiointiin projektin aikana. Käytettävyydestä vastuussa pitää kuitenkin olla projektitiimin jäsen, jolla on samaan aikaan tietämystä ohjelmistosta ja käytettävyydestä, tehtyjen suunnitelmien toteuttaminen on mahdollista.

6.1.2 Iteratiivisuus

Toisen periaatteen mukaiseen iteratiivisuuteen sisältyy, että tuoteversioista saadaan palautetta usein ja tuoteversiota kehitetään palautteen perusteella (Brhel et al. 2015). Haastatteluissa mainittiin, että usein suunnitelmista ja toteutuksesta saadaan palautetta liian myöhään, jolloin tehty työ voi olla turhaa ja toiminnallisuudet voidaan joutua toteuttamaan uudestaan. Lisäksi usein palaute tulee asiakkaalta, vaikka käyttäjät ovat itse parhaiten tietoisia tarpeistaan ja siitä, soveltuvatko suunnitellut tai tehdyt toiminnallisuudet heidän työtehtäviinsä. Projekteissa, joissa käyttäjiltä on kysytty palautetta iteraatioiden aikana, käyttäjien antama palaute on koettu hyvin hyödylliseksi sekä projektitiimin että asiakkaan puolesta. Sekä kirjallisuuden että haastatteluiden pohjalta voidaan siis todeta, että iteraatioissa on tärkeää saada palautetta myös käyttäjiltä.

Fox et al. (2008) mukaan käyttäjäkeskeisissä iteraatioissa suunnitellaan korjaukset projektin aikana havaittuihin ongelmiin, ja Salah et al. (2014) mukaan ketterän ohjelmistokehityksen iteraatioissa toteutetaan toiminnallisuudet suunnitelmien ja korjauksien perusteella. Käyttäjäkeskeisen ketterän ohjelmistokehityksen viitekehyksessä sekä suunnittelu että toiminnallisuuksien toteutus tehdään saman iteraation aikana, jotta yksikin tiimi riittää viitekehyksen noudattamiseen. Haastatteluissa ei noussut ongelmia tai haasteita siihen liittyen, että samassa iteraatiossa suunniteltaisiin ja toteutettaisiin iteraatioon valitut toiminnallisuudet. Sen sijaan, suunnittelun erottamista toteutuksesta pidettiin haasteellisena etenkin valmiin ohjelmiston asettamien rajoitusten vuoksi. Myös toteutuksen aloittamista iteraatioissa ennen suunnittelua ja jatkamista taas suunnittelun jälkeen ehdotettiin sopivan joihinkin projekteihin paremmin, sillä toteutuksen aloittamisen jälkeen suunnittelussa voidaan ottaa kantaa myös jo ilmenneisiin haasteisiin.

6.1.3 Suunnitelmien ja tuoteversioiden arviointi iteraatioissa

Kolmannen periaatteen mukaan iteraatioissa pidetään käytettävyyspalavereita ja katselmointeihin kutsutaan käyttäjiä. Haastatteluissa ehdotettiin, että joissain projekteissa riittää, että käyttäjäkeskeisyyttä lisätään projekteihin vain kutsumalla käyttäjät katselmointeihin. Sy (2007) ja Miller (2005) pitävät kuitenkin tärkeänä, että käyttäjät ovat arvioineet suunnitelmat ennen kuin toiminnallisuuksia aletaan toteuttaa. Syn (2007) ja Millerin (2005) lisäksi osa haastateltavista huomauttaa, että jos käyttäjät arvioivat pelkästään valmiit tuoteversiot, iteraation aikana toteutetut toiminnallisuudet voivat osoittautua turhiksi tai vääränlaisiksi. Käytännössä iteraation aikana toteutettaisiin projektitiimin arvauksen mukainen toiminnallisuus, katselmoinnissa käyttäjät arvioivat onko arvaus riittävä, ja jos toiminnallisuus ei vastaa käyttäjien tarpeisiin, oikea toiminnallisuus toteutetaan vasta

seuraavassa iteraatiossa, jossa pitäisi kehittää jo seuraavia toiminnallisuuksia edellisen korjaamisen sijaan.

Käyttäjäkeskeisessä ketterässä ohjelmistokehityksen viitekehyksessä käyttäjät arvioivat suunnitelmat iteraatiossa 0 ja käytettävyyspalaverissa, joten on suositeltavaa sisällyttää edes toinen niistä käyttäjäkeskeisiin projekteihin, jotta toiminnallisuuksien toteuttaminen useaan kertaan voidaan välttää. Jos projektissa ei voida järjestää käytettävyyspalavereita tai iteraatiota 0, edes käyttäjien kutsuminen katselmointiin mahdollistaa käytettävyysongelmien huomaamisen ennen valmiin ohjelmiston käyttöönottoa.

Kolmatta periaatetta muutettiin, jotta käyttäjäkeskeisiä ketteriä ohjelmistokehitysprojekteja voidaan toteuttaa yhdellä tiimillä. Kuten aiemmin on mainittu, kaikki haastateltavat kokivat, että yksikin tiimi riittää viitekehyksen noudattamiseen. Suurin osa piti yhtä tiimiä jopa parempana vaihtoehtona kuin kahta tiimiä, sillä yhdessä tiimissä kommunikointi on helpompaa ja tiimiläiset voivat luontevammin jakaa osaamista käytettävyydestä ja ohjelmiston rajoituksista toisilleen. Budvig et al. (2009), Sy (2007) ja Miller (2005) kannattavat irrallisia iteraatiovaiheita ja kahden tiimin käyttämistä, sillä heidän mukaansa kaksi erillistä tiimiä ja irralliset iteraatiovaiheet vähentävät kommunikaatio-ongelmia ja parantavat käytettävyyden suunnittelijoiden ja ohjelmistokehittäjien välistä yhteistyötä. Tämä ristiriita voi mahdollisesti johtua siitä, että kohdeyksikön projekteissa käytettävyysosaaminen ja tietämys ohjelmiston toiminnasta tulisi olla samalla henkilöllä, jotta suunnitelmat toimivat käytännössä valmiin ohjelmiston rajoituksista huolimatta. Lisäksi on mahdollista, että aiemmissa tutkimuksissa projektit ovat olleet suurempia tai käytettävyys on ollut niissä tärkeämmässä roolissa kuin kohdeyksikön projekteissa, sillä myös haastatteluissa nousi esiin, että kaksi tiimiä voisi olla parempi suurissa projekteissa, joissa työmäärä olisi luontevaa jakaa kahden tiimin kesken tai projekteissa, joissa käytettävyys on prioriteettina.

6.1.4 Käyttäjien osallistaminen

Neljännessä periaatteessa huomioidaan sidosryhmien osallistamisen tärkeys suunnittelu- ja toteutusvaiheissa projektin alusta alkaen (Brhel et al. 2015). Haastatteluissa tunnistettiin sekä asiakkaiden että käyttäjien osallistamisen hyödyt: vain käyttäjät voivat määrittää täyttävätkö toiminnallisuudet heidän tarpeensa ja asiakkaalla on näkemys kokonaisuudesta ja liiketoiminnan kannalta tärkeistä prosesseista. Kirjallisuudessa tätä työnjakoa ei korostettu, mutta Fox et al. (2008) mainitsee, että käyttäjäkeskeisessä suunnittelussa keskitytään nimensä mukaisesti käyttäjiin ja ketterässä ohjelmistokehityksessä on perinteisesti keskitytty vain asiakkaaseen. Haastatteluiden perusteella vaikut-

taa siltä, että ketterässä ohjelmistokehityksessä prosessien toimivuus koetaan tärkeämmäksi kuin käytettävyys, mikä voi olla syynä sille, ettei käyttäjiä ole aiemmin hyödynnetty projektien aikana. Kuten Lárusdóttir et al. (2017) mainitsevat, vain asiakkaan osallistaminen ei kuitenkaan takaa laadukasta ja käytettävää ohjelmistoa, mikä myönnettiin myös haastatteluissa.

Haastatteluissa mainittiin myös ongelmia, joita voi seurata käyttäjien osallistamisesta: Projektien hallinta ja aikataulutus vaikeutuu, kun projektiin osallistuvien henkilöiden määrä kasvaa. Lisäksi kustannukset kasvavat työmäärän lisääntyessä. Ongelmien välttämiseksi haastatteluissa ehdotettiin, että käyttäjillä ja asiakkaalla tulee olla selkeät roolit, eli käyttäjät voivat vaikuttaa käytettävyyteen, mutta asiakas tekee päätökset. Viitekehyksessä roolit on huomioitu, sillä käyttäjien palaute katselmoinnissa vaikuttaa visioon lopputuloksesta ja vain asiakas voi tehdä esimerkiksi uusia toiminnallisuuksia koskevat päätökset. Kustannusten laskemiseksi, Sy (2007) ehdottaa, että käyttäjiä kutsutaan katselmointeihin vasta keskivaiheen iteraatioista alkaen. Haastatteluissa kuitenkin pidettiin käyttäjien osallistamista alusta asti hyödyllisenä, sillä sen koettiin lisäävän käyttäjien luottamusta toimittajaa kohtaan ja lisäävän käyttäjien sitoutuneisuutta ohjelmistoon. Aikaa ja siten myös kustannuksia, voidaan säästää myös toteuttamalla arvioinnit etänä (Salah et al. 2014). Yhdessä kohdeyksikön projektissa käyttäjät saivat arvioida tuoteversioita myös etänä, ja se oli tässä projektissa toiminut. Vaikka osa käyttäjistä tulisikin paikanpäälle arvioimaan tuoteversioita, etämahdollisuuden tarjoaminen voi helpottaa myös käyttäjille työn aikatauluttamista, jolloin he voivat päästä helpommin osallistumaan katselmointiin.

Lievesley & Yee (2007) ja Fox et al. (2008) toteavat, että käyttäjien sijaan myös projektitiimin jäsenet voivat arvioida suunnitelmat ja tuoteversiot, jos käyttäjiä ei ole saatavilla. Viitekehyksessä projektitiimi arvioi muutokset suunnitelmissa käytettävyyspalaverin ja katselmoinnin välisenä aikana, eikä haastatteluissa noussut eriäviä mielipiteitä sitä kohtaan. Kuten Beyer (2010), Da Silva et al. (2012) ja Lárusdóttir et al. (2017), myös haastateltavat tunnistivat, että käyttäjät tietävät parhaiten, täyttävätkö toiminnallisuudet heidän tarpeensa ja ovatko he toiminnallisuuksiin tyytyväisiä. Haastatteluissa ehdotettiinkin, että kaikkien käyttäjien osallistamisen ja käyttäjien ”sijaisten” käyttämisen sijaan, katselmointeihin, käytettävyyspalavereihin ja iteraatioon 0 kutsuttaisiin suunnitelmien ja toteutusten kannalta oleellisista käyttäjäryhmistä edustajia. Käyttäjäryhmien edustajien valitseminen säästää kustannuksia, sillä se rajoittaa osallistuvien käyttäjien määrää, ja samalla varmistetaan, että toiminnallisuuksien kannalta olennaiset käyttäjäryhmät tulevat kuulluksi.

6.1.5 Prototyyppien hyödyntäminen

Viidennessä periaatteessa korostetaan, että sidosryhmien kanssa tulee kommunikoida esimerkiksi prototyyppien tai rautalankamallien kautta (Salah et al. 2014; Brhel et al. 2015). Haastateltavat pitivät prototyypeistä erityisesti siksi, että niiden avulla voidaan välttää väärinkäsityksiä, kun projektitiimi, asiakas ja käyttäjä näkevät suunnitelmat konkreettisessa muodossa. Myös Da Silva et al. (2012) ja Lárusdóttir et al. (2017) mainitsevat, että prototyypin avulla suunnitelmat voidaan selittää käyttäjille mahdollisimman vaivattomasti. Lisäksi haastatteluissa mainittiin, että prototyypit helpottavat projektitiimin töitä, sillä konkreettisten suunnitelmien pohjalta toiminnallisuuksien toteuttaminen on helpompaa.

Kirjallisuudessa ei huomioitu, että myös prototyypin tekeminen vie aikaa. Sen sijaan, haastatteluissa ajan asettamat haasteet mainittiin useasti, ja osa ei hyödyntäisi prototyyppejä iteraation 0 aikana ollenkaan. Lisäksi toisinaan toiminnallisuudet voidaan toteuttaa samassa ajassa suoraan valmiiseen järjestelmään, jolloin prototyypin tekeminen olisi turha välivaihe. Prototyypin käyttöä kannattaakin harkita projekti- ja jopa toiminnallisuuskohtaisesti, ja kuten haastatteluissa nousi esiin, vain tärkeimmistä toiminnallisuuksista kannattaa tehdä prototyyppi.

6.2 Viitekehyksen kehittäminen

Haastatteluiden pohjalta viitekehystä kehitetään joustavammaksi ja aikaisempaa sopivammaksi erilaisiin projekteihin. Taulukkoon 16 on koottu kaikki viitekehyksestä havaitut ongelmat ja haastatteluista sekä kirjallisuudesta nousseet ratkaisut näihin ongelmiin. Kirjallisuudesta poimitut ratkaisut on merkitty taulukkoon kursivoidulla tekstillä. Lisäksi muokatussa viitekehyksessä (kuva 9) huomioidut kehitysehdotukset on merkitty tummennetulla tekstillä.

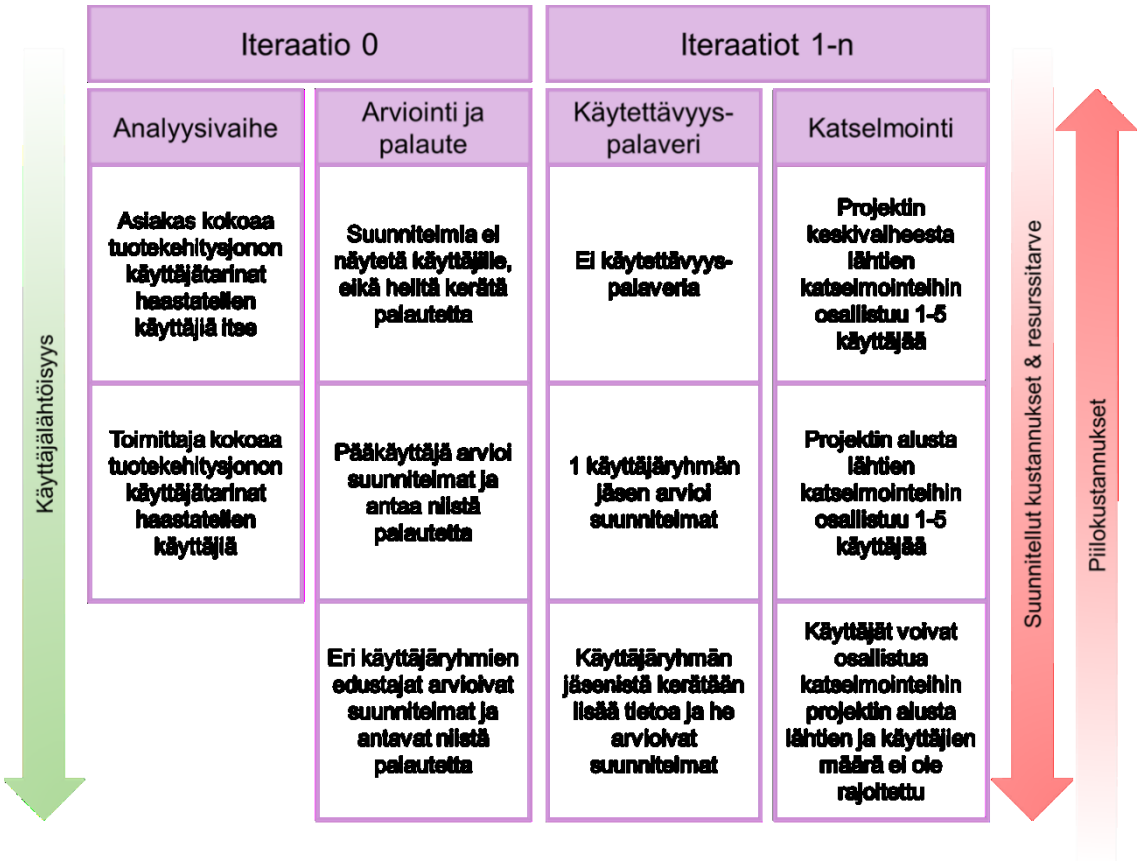
Taulukko 16. *Yhteenveto viitekehyksen haasteista ja kehitysehdotuksista*

Aihealue	Haasteet	Kehitysehdotukset
Viitekehys yleisesti	Työmäärän kasvu ja lisäkustannukset, asiakas ei halua maksaa käyttäjäkeskeisyydestä	Työmäärän kasvun huomioiminen ja lisäajan varaaminen myyntivaiheessa. **Viitekehyksestä tehdään useita, kevyempiä versioita.** **Asiakkaalle annetaan vastuu iteraation 0 analyysivaiheesta.** **Käytettävyyspalavereissa projektitiimi tekee suunnitelmat kartoittamatta käyttäjien tarpeita ja kysyy valmiista suunnitelmista palautetta käyttäjiltä.** **Käyttäjät kutsutaan vain katselmointiin iteraation aikana, ei järjestetä käytettävyyspalaveria.** **Tunnistetaan tärkeimmät toiminnallisuudet.** Käyttäjät voivat arvioida toiminnallisuuksia ja antaa niistä kommentteja etänä (Salah et al. 2014, Lárusdóttir et al. 2017)
	Oikeiden käyttäjien löytäminen	Määritellään käyttäjäryhmät. Annetaan asiakkaalle vastuu käyttäjien valinnasta.
	Kokonaisuuden hallinnan vaikeutuminen	Asiakkaalle ja käyttäjille tehdään selkeä vastuunjako.

	Järjestelmän asettamat rajoitteet	Järjestelmän esitellään käyttäjille jo ennen iteraatiota 0. Projektitiimillä on ymmärrys järjestelmän rajoituksista.
Iteraatio 0	Käyttäjät haluavat uuden järjestelmän olevan samanlainen kuin edellinen	Ammattilaisten käytettävyyden suunnittelijoiden hyödyntäminen, sillä he osaavat tunnistaa tarpeet toiveiden takana (Da Silva et al. 2012)
	Etukäteen tehdyt, liian tarkat suunnitelmat rajoittavat ketteryyttä	Iteraatiossa 0 keskitytään vain käyttäjän tarpeisiin ja toiminnallisuuksiin, joilla tarpeet voidaan täyttää.
Käytettävyyspalaveri	Projektit ovat keskenään hyvin erilaisia	Käytettävyyspalaverin ajankohdan joustavuus: projektista riippuen käytettävyyspalaveri voi olla heti suunnittelupalaverin jälkeen tai projektissa voi olla vähän kehitystä jo ennen käytettävyyspalaveria.
Katselmointi	Käyttäjien runsas lukumäärä	**Rajoitetaan osallistuvien käyttäjien lukumäärää.**
Prototyyppi	Iteraatio 0 liian varhainen vaihe prototyypille	Prototyyppi tehdään vasta käytettävyyspalaverissa.
	Prototyypin tekeminen vie aikaa	Prototyyppi tehdään vain tärkeimpien toiminnallisuuksien osalta. Pienten kustomointien osalta muutokset tehdään suoraan järjestelmään.

Yhden tiimin projektit ja roolit	Ulkopuolisen käytettävyyden ammattilaisen ratkaisut eivät välttämättä toimi valmiissa järjestelmässä	Tietämys käytettävyydestä ja järjestelmän rajoitteista tulee olla samalla henkilöllä.
	Projektitiimiläisillä ei ole tällä hetkellä käytettävyysosaamista	Ulkopuolinen käytettävyyden ammattilainen kouluttaa käytettävyyttä projektin aikana.

Viitekehystä on muokattu joustavammaksi (kuva 9) haastatteluissa ehdotettujen kehitysehdotusten pohjalta. Viitekehyksen runko (kuva 8) pysyy muuttumattomana, mutta kuvaa 9 voidaan hyödyntää esimerkiksi myyntitilanteessa, jolloin asiakkaan kanssa voidaan valita heidän projektiinsa sopivat vaihtoehdot iteraation 0 analyysi-, arviointi- ja palautevaiheille, sekä muiden iteraatioiden käytettävyyspalavereille ja katselmoinneille. Vaihtoehdot vaihtelevat vaiheen poisjättämisestä tai asiakkaan itsenäisestä toteutuksesta, useiden käyttäjien osallistamiseen jokaisessa vaiheessa projektin alusta loppuun asti. Kuten kuvasta 9 on nähtävissä, osa vaihtoehdoista on käyttäjälähtöisempiä kuin toiset, ja käyttäjälähtöisemmissä vaihtoehdoissa suunnittellut kustannukset ja resurssitarve kasvaa. Kustannuksissa on myös huomioitava, että piilokustannukset pienenevät, sillä ongelmien korjaaminen on sitä kalliimpaa, mitä myöhemmin ne huomataan. Jos projekteihin lisätään tarpeeksi aikaisin käyttäjälähtöisyyttä, ongelmat on mahdollista huomata jo ennen toteutusta.



Kuva 9. *Viitekehyksen mukauttaminen erilaisiin projekteihin*

Viitekehyksen voi mukauttaa esimerkiksi pieneen projektiin, jossa asiakas ei tiedä tarkasti organisaationsa tarpeita. Tällaiseen projektiin voisi sopia toteutus, jossa toimittaja toteuttaa iteraation 0 analyysivaiheen, ja jossa katselmointeihin kutsutaan muutama käyttäjä antamaan valmiista tuoteversioista palautetta heti projektin alusta lähtien. Kustannuksien säästämiseksi iteraation 0 muita vaiheita ja käytettävyyspalaveria ei toteuteta. Tässä projektissa toimittaja tutustuu organisaatioon, sen käyttäjiin ja heidän työtehtäviinsä ja tarpeisiinsa analyysivaiheen aikana, sekä tekee käyttäjistä kerättyjen tietojen pohjalta käyttäjätarinat tuotekehitysjonoon. Projektitiimi tekee suunnitelmat toteutuksesta itsenäisesti, eikä hyödynnä käyttäjiä suunnitelmien arvioinnissa iteraation 0 aikana. Muissa iteraatioissa ei järjestetä käytettävyyspalavereita, vaan projektitiimi tekee iteraatiossa 0 keräämiensä tietojen pohjalta arvauksia toiminnallisuuksista ja toteuttaa ne. Katselmoinnissa käyttäjät kertovat, täyttävätkö toteutetut toiminnallisuudet heidän tarpeensa ja ovatko he tyytyväisiä toteutukseen. Tarvittaessa projektitiimi korjaa toiminnallisuudet seuraavassa iteraatiossa, mikä lisää piilokustannuksia.

Toisen ääripään esimerkki voidaan muodostaa suuresta projektista, jossa asiakas tietää organisaation tarpeet. Koska asiakas on itsenäinen, toimittajaa ei tarvita analyysivaihee-

seen, vaan asiakas selvittää itse käyttäjien tarpeet ja muodostaa käyttäjätarinat tuotekehitysjonoon. Tarvittaessa toimittaja antaa tiiviin ohjeistuksen käyttäjätarinoiden kokoamiseen. Myöhemmin iteraatiossa 0 käyttäjät arvioivat toimittajan tekemät suunnitelmat ja toimittaja tekee niihin tarvittaessa muutoksia palautteen pohjalta. Käytettävyyspalavereita ja katselmointeja järjestetään iteraatioissa alusta lähtien. Käytettävyyspalaverissa kerätään käyttäjistä ja heidän työtehtävistää lisää tietoa iteraation aikana toteutettavia toiminnallisuuksia varten, jotta projektitiimillä on mahdollisimman hyvä tuntemus käyttäjien tarpeista. Käytettävyyspalaveriin kutsutaan toiminnallisuuksien kannalta oleellisten käyttäjäryhmien jäseniä, mutta katselmointeihin käyttäjien määrää ei ole rajoitettu, jotta myös ne käyttäjät, jotka eivät osallistuneet käytettävyyspalaveriin, pääsevät antamaan toiminnallisuuksista palautetta, kertomaan ideoistaan, näkemään projektin etenemisen, tutustumaan ohjelmistoon ennen sen käyttöönottoa ja samalla heille voidaan osoittaa, että käyttäjiä on kuunneltu ohjelmistoa tehdessä. Koska käyttäjät ovat arvioineet suunnitelmat käytettävyyspalaverissa, suurimmat ongelmat on huomattu jo sen aikana ja suunnitelmat voidaan korjata jo ennen toteutusta. Katselmoinnin pohjalta saatu palaute sisältää todennäköisemmin pieniä viilauksia ja myöhemmissä iteraatioissa hyödyllisiä ideoita. Ideoiden suhteen on kuitenkin hyvä huomioida, että käyttäjiltä saadut ehdotukset tulee olla asiakkaan hyväksymiä ennen niiden lisäämistä tuotekehitysjonoon, jotta työmäärä ei kasva liiaksi ja jotta tuotteen kehitys pysyy asiakkaan liiketoiminnan kannalta oleellisena.

Prototyyppien käyttöä ei ole mainittu kuvassa 9, sillä projektitiimi valitsee itse, milloin prototyyppejä käytetään ja mitkä toiminnallisuudet ovat tarpeeksi tärkeitä. Mikäli asiakas päättää prototyyppien käytöstä, se voi sitoa projektitiimin tekemään prototyyppejä liian aikaisin, kun suunnitelmat eivät ole vielä tarpeeksi valmiita. Lisäksi liian aikaisessa vaiheessa tehdyt prototyypit voivat määrittää tulevan toteutuksen liian tarkasti jo ennen kuin projektitiimillä on tarpeeksi tietämystä siitä, miten ohjelmisto kannattaa toteuttaa.

Viitekehystä suositellaan noudatettavaksi yhdellä tiimillä, mutta tarvittaessa projektin toteutukseen voi osallistua kaksi tiimiä, jos asiakas kokee sen tarpeelliseksi projektin koon tai käytettävyyden tärkeyden vuoksi. Projekteissa, joissa kustomoidaan valmista ohjelmistoa, käytettävyysosaamisen ja tietämyksen ohjelmiston rajoituksista tulee olla samalla henkilöllä, jotta suunnitelmat on mahdollista toteuttaa käytännössä. Jos tiimissä ei vielä ole käytettävyysosaamista, tiimiä voidaan kouluttaa ulkopuolisen käytettävyyden ammattilaisen avulla.

7. YHTEENVETO

7.1 Päätulokset

Tutkimuksen aikana selvitettiin, kuinka ketterää ohjelmistokehitystä voidaan toteuttaa käyttäjäkeskeisemmin. Ensimmäiseksi vastattiin kysymykseen ”Mitkä käytännöt tukevat käyttäjäkeskeisyyden sovittamista ketterään ohjelmistokehitykseen?”. Käyttäjäkeskeisyyden ja ketteryyden yhteensovittamista tukevat käytännöt ovat erillisen suunnitteluvaiheen pitäminen projektin alussa, tuotteen iteratiivinen toteuttaminen, käytettävyyspalaverien järjestäminen ja käyttäjien kutsuminen katselmointeihin, sidosryhmien aktiivinen osallistaminen ja tuotosten kommunikointi helposti ymmärrettävässä muodossa. Nämä käytännöt löytyivät suurimmaksi osaksi suoraan aiemmasta kirjallisuudesta, sillä (Brhel et al. 2015) ovat koonneet ne kirjallisuuskatsauksessaan. Kuten aiemmin on mainittu, ainoastaan yhtä periaatetta muokattiin, jotta viitekehys on noudatettavissa ilman toista tiimiä.

Toiseen kysymykseen, ”Miten käytettävyydestä vastaavan tiimin puuttuminen vaikuttaa viitekehykseen?”, vastaus muodostettiin soveltaen Syn (2007) viitekehystä, jossa käytettävyyden suunnittelu ja arviointi toteutetaan eri iteraatiossa kuin ohjelmistokehitys. Toisen tiimin puuttuessa, suunnittelu ja ohjelmistokehitys pitää toteuttaa samassa iteraatiossa, jonka vuoksi ohjelmistokehityksen iteraatioon lisätään käytettävyyspalaveri, jossa tehdään iteraatiokohtaiset käytettävyyssuunnitelmat, ja asiakkaiden lisäksi myös käyttäjät kutsutaan katselmointeihin arvioimaan iteraation aikana toteutetut toiminnallisuudet. Tutkimuksessa haastatellut henkilöt pitivät mahdollisena noudattaa uutta viitekehystä yhden tiimin voimin, ja suurin osa piti sitä myös parempana vaihtoehtona kuin erillisen käytettävyystiimin ottamista mukaan projekteihin.

Kolmannen kysymyksen, ”Miten projektien yksilöllisyys vaikuttaa viitekehykseen?”, vastaus koostettiin haastatteluiden pohjalta, ja se on tiivistetty kuvaan 9. Haastatteluissa nousi useasti esiin, että viitekehyksen tulee olla joustava sopiakseen erilaisiin projekteihin. Osassa haastatteluista mainittiin myös konkreettisia ehdotuksia: itsenäinen asiakas voi ottaa vastuun iteraation 0 analyysivaiheesta, käytettävyyspalaveria ei välttämättä kannatta järjestää pienissä projekteissa ja toisaalta isoissa projekteissa käyttäjiä voidaan osallistaa jokaisessa iteraatiossa sekä katselmointiin että käytettävyyspalaveriin.

Haastatteluiden ja kirjallisuuden pohjalta vaikuttaa siltä, että käyttäjäkeskeisyyttä voidaan lisätä ketterän ohjelmistokehityksen projekteihin jo pelkästään kutsumalla käyttäjät mukaan katselmointiin. Vaikka tämä käytäntö ei poistakaan riskiä toiminnallisuuksien käytettävyysongelmista ja niiden uudelleen toteuttamisesta, käytettävyysongelmat huomataan ennen varsinaista käyttöönottoa ja ne on mahdollista korjata seuraavan iteraation aikana. Mitä useammat suunnitelmat käyttäjät saavat arvioida, sitä pienempi riski on käytettävyysongelmien aiheuttamille lisäkustannuksille, sillä käyttäjien osallistaminen ajoissa mahdollistaa käytettävyysongelmien huomaamisen ennen toiminnallisuuksien toteuttamista. Lyhyesti tiivistettynä, ketterää ohjelmistokehitystä voidaan toteuttaa käyttäjäkeskeisemmin osallistamalla käyttäjiä enemmän suunnitelmien ja valmiiden toteutusten arviointiin.

7.2 Suositukset käytäntöön

Tämän tutkimuksen aikana luotu viitekehys tarjoaa aiempia viitekehyksiä (Miller 2005; Sy 2007; Fox et al. 2008) paremmat mahdollisuudet toteuttaa käyttäjäkeskeistä ketterää ohjelmistokehitystä myös sellaisissa projekteissa, joissa liiketoimintaprosessien toimivuus on tärkeämpää kuin käytettävyys tai joissa ei ole mahdollista hyödyntää kahta tiimiä. Aiemmat viitekehykset on luotu käytettävyysammattilaisten näkökulmasta: miten käyttäjäkeskeisyyttä voidaan paremmin toteuttaa ketterässä ohjelmistokehityksessä. Tässä tutkimuksessa aihetta lähestytään ketterän ohjelmistokehityksen näkökulmasta: miten ketterää ohjelmistokehitystä voidaan toteuttaa käyttäjäkeskeisemmin. Uusi näkökulma mahdollistaa käyttäjäkeskeisyyden lisäämisen asteittain, projektin tarpeiden mukaan, kun taas edellisissä tutkimuksissa ei ole otettu huomioon projektien yksilöllisiä tarpeita.

Viitekehyksen avulla yrityksillä on mahdollisuus lisätä käyttäjäkeskeisyyttä ketterissä projekteissa, vaikka yrityksillä ei olisi aiempaa kokemusta käyttäjäkeskeisyydestä tai omaa käytettävyystiimiä. Yksinkertaisimmillaan käyttäjäkeskeisyys on käyttäjien osallistamista, mihin ei välttämättä tarvita suuria lisäresursseja, joten käyttäjäkeskeisyyden lisääminen on mahdollista, vaikka asiakas ei olisi valmis maksamaan käyttäjäkeskeisyydestä. Koska käytettävyysongelmien korjaaminen käyttöönoton jälkeen on kallista (Ferreira et al. 2007b), ja ne voivat jopa johtaa siihen, etteivät käyttäjät halua käyttää ohjelmistoa (Kuusinen & Väänänen-Vainio-Mattila 2012; Da Silva et al. 2018), käyttäjäkeskeisyys säästää projektien kustannuksia mahdollistamalla käytettävyysongelmien huomaamisen ajoissa, jopa ennen toiminnallisuuksien toteutusta.

Tämän tutkimuksen pohjalta voidaan suositella, että myös ne yritykset, joilla ei ole tarpeeksi resursseja kahden tiimin sitouttamiseen yhteen projektiin tai joissa ei ole aiempaa käytettävyysosaamista, lisäävät ketterään ohjelmistokehitykseen käyttäjälähtöisyyttä heille ja heidän asiakkailleen sopivalla tasolla. Käyttäjiä osallistamalla voidaan selvittää käyttäjien todelliset tarpeet, sillä he itse tuntevat parhaiten omat tarpeensa, työtehtävänsä ja mielipiteensä toiminnallisuuksista. Pienissä projekteissa voi riittää, että käyttäjät kutsutaan katselmointeihin antamaan palautetta toteutetuista toiminnallisuuksista, jotta käytettävyysongelmat voidaan korjata seuraavassa iteraatiossa. Lisäksi käyttäjien osallistaminen lisää käyttäjien sitoutuneisuutta ohjelmiston käyttöön ja luottamusta ohjelmiston toimittajaan, kun käyttäjät näkevät palautteensa merkityksen. Mitä enemmän käyttäjiä osallistetaan projektin aikana, sitä suuremmat ovat osallistamisen hyödyt. Esimerkiksi toiminnallisuuksien arviointi käyttäjien toimesta mahdollistaa käytettävyysongelmien huomaamisen jo ennen toteutusta, jolloin voidaan välttää toiminnallisuuksien uudelleen toteuttaminen seuraavassa iteraatiossa ja säästää kustannuksia.

7.3 Työn arviointi

Laadullisen tutkimuksen luotettavuuden osoittaminen on haastavaa, sillä tutkija voi vaikuttaa tutkimuksen tuloksiin monin eri tavoin. Esimerkiksi haastattelututkimuksissa tutkijan muodostamat haastattelukysymykset voivat johdatella haastateltavia vastaamaan kysymyksiin tietyllä tavalla. Shentonin (2004) ja Korstjensin & Moserin (2018) mukaan Lincolnin & Guban (1985) kokoama kriteeristö on tunnetuin laadullisen tutkimuksen arvioinnin kriteeristö. He ovat soveltaneet määrällisen tutkimuksen kriteerejä, pätevyyttä (*engl. internal validity*), yleistettävyyttä (*engl. generalisability*), luotettavuutta (*engl. reliability*) ja neutraaliutta (*engl. objectivity*), muokaten niitä paremmin sopivaksi laadulliseen tutkimukseen. Lincolnin ja Guban kriteerit ovat seuraavat:

- Uskottavuus (*engl. credibility*), eli kuinka luotettavia tutkimuksen tulokset ovat (Korstjens & Moser 2018) ja vastaavatko tulokset todellisuutta (Shenton 2004).
- Siirrettävyys (*engl. transferability*), eli voidaanko tutkimuksen tulokset sovittaa myös muihin tilanteisiin (Korstjens & Moser 2018).
- Totuudellisuus (*engl. debendability*), eli kuinka hyvin tutkimus on toistettavissa (Shenton 2004; Korstjens & Moser 2018).
- Vahvistettavuus (*engl. confirmability*), eli kuinka hyvin johtopäätökset pohjautuvat tutkimuksen aikana kerättyyn aineistoon (Shenton 2004; Korstjens & Moser 2018).

Uskottavuus on laadullisen tutkimuksen luotettavuuden kannalta tärkein kriteeri (Lincoln & Guba 1985). Shentonin (2004) mukaan tutkimuksen uskottavuuden voi varmistaa muun muassa mittaamalla tutkittavaa ilmiötä oikeilla tavoilla, tutustumalla tutkimuksen kohteeseen ennen aineiston keräämistä, valitsemalla tutkimukseen osallistuvat henkilöt sattumanvaraisesti, yhdistämällä erilaisia menetelmiä tai tietolähteitä ja vertaamalla tutkimuksen tuloksia aiempaan kirjallisuuteen. Lisäksi uskottavuuteen vaikuttaa tutkimukseen osallistuvien rehellisyys.

Tässä tutkimuksessa tutkittavaa ilmiötä mitattiin haastatteluiden muodossa, joten haastattelukysymysten arviointi on oleellista tutkimuksen uskottavuuden kannalta. Haastatteluiden tarkoitus oli arvioida kirjallisuuden pohjalta muodostetun käyttäjäkeskeisen ohjelmistosuunnittelun ja ketterän ohjelmistokehityksen yhteensovittamisen viitekehyksen soveltuvuutta käytäntöön ja yhden tiimin projekteihin. Haastateltavilta kysyttiin, kokisivatko he mahdolliseksi noudattaa viitekehystä yhden tiimin projekteissa ja mielipiteitä viitekehyksen haasteista ja hyödyistä. Lisäksi haastateltavilta kerättiin kehitysehdotuksia. Mielipiteiden ja kehitysehdotusten pohjalta tutkimuksessa muodostettiin käsitys viitekehyksen soveltuvuudesta käytäntöön. Haastateltavien vastausten luotettavuutta rajoittaa, että he eivät päässeet kokeilemaan viitekehystä käytännössä, jolloin heidän vastauksensa perustuvat oletuksiin. Haastateltavien vastausten rehellisyys pyrittiin varmistamaan antamalla heille mahdollisuus kieltäytyä haastattelusta, ja korostamalla heille haastattelun aikana, että myös kriittinen palaute on tutkimuksen kannalta erittäin tärkeää.

Ennen varsinaisen aineiston keräämistä, tutkittavaan yksikköön, projektikäytäntöihin ja nykytilanteeseen tutustuttiin tekemällä lyhyitä haastatteluita yksikön työntekijöille. Näiden haastatteluiden tarkoitus oli muodostaa parempi ymmärrys tutkittavasta ilmiöstä ja kohdeyksiköstä. Tutkimuksen kohteeseen tutustuminen on uskottavuuden kannalta tärkeää, koska se lisää haastateltavien luottamusta haastattelijaan. Tutkimuksen kohteen ei tulisi kuitenkaan olla liian tuttu tutkijalle, jotta tutkijasta ei ole puolueellinen. (Shenton 2004). Tämän tutkimuksen tapauksessa tutkimuksen uskottavuutta rajoittaa tutkijan työsuhde kohdeyksikköön, sillä se voi lisätä tutkijan puolueellisuutta. Työsuhteen vaikutusta vähentää kuitenkin sen tuoreus ja se, ettei tutkija valinnut itse haastatteluun osallistuneita henkilöitä.

Tutkimuksessa ei yhdistelty useita tutkimusmenetelmiä, mutta tietolähteinä käytettiin henkilöitä useista eri projekteista ja eri työtehtävistä. Eri työtehtävissä työskentelevien henkilöiden haastatteleminen mahdollisti erilaisten näkökulmien käsittelemisen ja eri

projekteissa oli erilaisia projektikäytänteitä, mikä lisäsi tulosten monipuolisuutta. Projektikäytänteiden eroihin sisältyi, että osassa projekteista oli osallistettu loppukäyttäjiä aiemminkin, mutta osassa projekteista käyttäjiä ei hyödynnetty projektin aikana. Aineiston monipuolisuus lisää aineiston uskottavuutta ja saatuja tuloksia arvioitiin kriittisesti vertaamalla niitä aiempaan kirjallisuuteen luvussa 6. Tämän tutkimuksen tulosten ja aiemman kirjallisuuden välillä oli joitakin ristiriitoja, esimerkiksi yhden tai kahden tiimin hyödyistä ja haasteista. Suurimmaksi osaksi tulokset olivat kuitenkin samassa linjassa aiemman kirjallisuuden kanssa, mikä lisää tutkimuksen uskottavuutta.

Tutkimuksen siirrettävyyttä voidaan vahvistaa kertomalla tutkimuksessa, mitkä ovat tutkimuksen rajaukset, millainen tutkimuksen kohde on ja mikä on tutkimuksen kohteen nykytilanne, jotta tutkimuksen lukija voi arvioida, onko tutkimus sovellettavissa hänen sovelluskohteeseensa (Shenton 2004). Tutkimuksen rajauksista muun muassa ketterään ohjelmistokehitykseen ja yrityksiin, joissa ei ole aiempaa kokemusta käyttäjäkeskeisestä ohjelmistosuunnittelusta, kerrotaan jo luvussa 1. Luvussa 4 on kuvailtu kohdeorganisaatio ja -yksikkö yleisellä tasolla, joka mahdollistaa lukijalle yleiskuvan muodostumisen tutkimustilanteesta. Lisäksi luvussa 5 on kuvattu kohdeyksikön projektien nykytila, jonka perusteella lukija voi määrittää, onko hänen sovelluskohteessaan samankaltainen tilanne vai poikkeaako se niin vahvasti kohdeyksikön tilanteesta, että tämän tutkimuksen tulokset eivät päde hänen sovelluskohteeseensa.

Jos tutkimus on uskottava, se on usein myös totuudellinen (Lincoln & Guba 1985). Tutkimuksen totuudellisuuden voi osoittaa kuvaamalla tutkimuksen toteutuksen tarpeeksi yksityiskohtaisesti, jotta tutkimus on mahdollista toistaa. Tutkimuksen toteutuksesta tulee kertoa tutkimussuunnitelma ja sen toteutus, kuinka aineisto on kerätty sekä arvioida valitun tutkimustavan sopivuutta tutkimukseen. (Shenton 2004). Tämän tutkimuksen luvussa 1 on kuvattu, mitä tutkimuksen aikana aiotaan tehdä ja mikä on tutkimuksen tavoite. Luvussa 4 kuvataan, kuinka tutkimus toteutettiin käytännössä, sisältäen listauksen tutkimuskysymyksistä, taulukon haastatteluun osallistuneista henkilöistä ja kuvauksen haastatteluiden toteutuksesta. Lisäksi luvussa 4 perustellaan, miksi valittu tutkimustapa sopii tähän tutkimukseen.

Tutkimuksen vahvistettavuuteen liittyy hyvin vahvasti tutkijan rooliin tutkimuksen teossa ja vahvistettavuutta voidaan arvioida sen perusteella, kuinka hyvin tutkija myöntää omat uskomuksensa, taipumuksensa toimia tietyllä tavalla ja rajoitteensa (Shenton 2004). Vahvistettavuuden, kuten totuudellisuudenkin, osalta on tärkeää, että tutkija perustelee tekemänsä valinnat (Shenton 2004), jolloin tutkija joutuu pohtimaan tekemiensä valintojen syitä. Tämän tutkimuksen suurimmat riskit, että tutkija on vaikuttanut tutkimuksen

tuloksiin, ovat olleet seuraavat:

- Haastateltavien johdattelu tutkijan tekemien haastattelukysymysten ja tarkentavien jatkokysymysten kautta
- Haastateltavien vastauksiin vaikuttaminen haastattelutilanteessa tutkijan omien asenteiden ja oletusten kautta
- Viitekehyksen esittelyn aikana tapahtuva johdattelu ja tutkijan omien mielipiteiden esiin nouseminen
- Tulosten ryhmittely ja kokoaminen haastatteluiden pohjalta: tutkijalle tärkeisiin asioihin keskittyminen ja vähemmän tärkeiden asioiden sivuuttaminen

Riskien minimoimiseksi tutkimuksen aikana on tietoisesti pyritty tekemään tutkimuskysymykset, jatkokysymykset ja viitekehyksen esittely mahdollisimman neutraalisti. Haastattelutilanteessa on siis vältetty tutkijan omien mielipiteiden esiin tuomista ja keskitytty haastateltavien vastausten yksityiskohtien selvittämiseen jatkokysymysten avulla. Tuloksissa on myös mainittu kaikki haastatteluissa esiin nousseet asiat, jotta lukijalla on mahdollisuus nähdä kaikki haastattelun tulokset. Kuitenkin tutkijan mielestä vähemmän tärkeät aiheet on käsitelty pintapuoleisemmin kuin muut aiheet.

7.4 Jatkotutkimusaiheet

Seuraavissa tutkimuksissa tätä viitekehystä tulisi testata käytännössä erilaisissa projekteissa ja useissa yrityksissä, jolloin viitekehystä on mahdollista kehittää eteenpäin aitojen kokemusten pohjalta ja paremmin soveltuvaksi kaikentyyppisiin projekteihin, myös sellaisiin, joissa valmis ohjelmisto ei aseta rajoitteita käytettävyyden suunnittelulle. Jatkossa voisi siis tutkia, *kuinka käyttäjäkeskeisyyttä voidaan lisätä ketterissä ohjelmistokehitysprojekteissa, joissa tuote tehdään alusta asti*, ja *kuinka käyttäjäkeskeisen ohjelmistosuunnittelun ja ketterän ohjelmistokehityksen yhteensovittamisen viitekehys soveltuu käytännössä ketteriin ohjelmistokehitysprojekteihin*.

Tässä tutkimuksessa ei selvitetty, kuinka kauan analyysivaiheen, käytettävyyspalaverin tai katselmoinnin tulee kestää, vaikka Hodgetts (2005) on painottanut, että suunnittelulle pitää asettaa aikarajat, joten sekin jää tulevien tutkimusten selvitettäväksi. Myös muiden yksityiskohtien, kuten ketterään käyttäjäkeskeiseen suunnitteluun parhaiten soveltuvien menetelmien, selvittämiselle on kysyntää (Lárusdóttir et al. 2013), joten nekin tarjoavat mahdollisuuden jatkotutkimukselle. Jatkotutkimuksessa voitaisiin siis selvittää, *mitkä ovat parhaat menetelmät käyttäjäkeskeisessä ja ketterässä ohjelmistokehityksessä*.

Käyttäjäkeskeisyydestä aiheutuvia kustannuksia ja säästöjä, sekä sen vaikutuksia riskeihin käsitellään tässä tutkimuksessa hyvin pintapuoleisesti. Niiden selvittäminen jatkotutkimuksessa mahdollistaa uskottavat perustelut käyttäjäkeskeisyyden yhteensovittamiselle ketterään ohjelmistokehitykseen. Jatkotutkimuksessa voidaan verrata käyttäjäkeskeisen ja ketterän ohjelmistokehitysprojektin suunniteltuja ja toteutuneita kustannuksia ketterän ohjelmistokehityksen suunniteltuihin ja toteutuneisiin kustannuksiin, ja selvittää, *voidaanko ketterän ohjelmistokehityksen kustannuksia pienentää käyttäjäkeskeisyyttä lisäämällä.* Kuten haastatteluissa mainittiin, käyttäjäkeskeisyyden lisääminen kannattaa, jos se alentaa kustannuksia.

LÄHTEET

Agile Alliance. (2001). Manifesto for Agile Software Development. WWW-julkaisu. Saatavilla: https://agilemanifesto.org/ (Viitattu 31.5.2019).

Armitage, J. (2004). Are Agile Methods Good for Design? Interactions. Vol. 11(1). pp. 14-23.

Artto, K., Martinsuo, M., Kujala, J. (2006). Projektiliiketoiminta. Helsinki: WSOY.

Ashraf, S., Aftab, S. (2017). IScrum: An improved scrum process model. I. J. Modern Education and Computer Science. Vol. 8. pp. 16-24.

Brhel, M., Meth, H., Maedche, A., Werder, K. (2015). Exploring principles of user-centered agile software development: A literature review. Information and software technology. Vol. 61. pp. 163-181.

Budvig, M., Jeong, S., Kelkar, K. (2009). When user experience met agile: A case study. CHI.

Cockburn, A., Highsmith, J. (2001). Agile software development: The people factor. IEEE Computer. Vol. 34(11). pp. 131-133.

Conboy, K. (2009). Agility from first principles: Reconstructuring the concept of agility in information systems management. Information Systems Research. Vol. 20(3). pp. 329-354.

Cooper, A. (1999). The inmates are running the asylum: Why high-tech products drive us crazy and how to restore the sanity. Indianapolis, Indiana: SAMS.

Da Silva, T. S., Martin, A., Maurer, F., Silveira, M. S. (2011). User-centered design and agile methods: A systematic review. Agile 2011. pp. 77-86.

Da Silva, T. S., Silveira, M. S., Maurer, F., Hellman, T. (2012). User experience design and agile development: From theory to practice. Journal of software engineering and applications. Vol. 5. pp. 743-751.

Da Silva, T. S., Silveira, M. S., Maurer, F., Silveira, F. F. (2018). The evolution of agile UXD. Information and Software Technology. Vol. 102. pp. 1-5.

Detweiler, M. (2007). Managing UCD within agile projects. Interactions. Vol. 14(3). pp. 40-42.

Dingsøyr, T., Nerur, S., Balijepally, V., Moe, N. B. (2012). A decade of agile methodologies: Towards explaining agile software development. The journal of Systems and Software. Vol. 85. pp. 1213-1221.

Ferreira, J., Noble, J., Biddle, R. (2007a). Interaction designers on eXtreme programming teams: Two case studies from the real world. Proceedings of the 5th New Zealand Computer Science Research Student Conference.

Ferreira, J., Noble, J., Biddle, R. (2007b). Up-front interaction design in agile development. Proceedings of the 8th International Conference on Agile Processes in Software Engineering and Extreme Programming. pp. 9-16.

Fox, D., Sillito, J., Maurer, F. (2008). Agile methods and user-centered design: how these two methodologies are being successfully integrated in industry. Agile 2008 Conference.

Gould, J. D., Boies, S. J., Ukelson, J. (1997). How to design usable systems. In Helander, M., Landauer, T. K., Prabhu, P. Handbook of Human-Computer Interaction. Amsterdam: Elsevier Science B. V.

Gulliksen, J., Göransson, B., Boivie, I., Blomkvist, S., Persson, J., Cajander, Å. (2003). Key principles for user-centred systems design. Behaviour & Information Technology. Vol. 22(6). pp. 397-409.

Highsmith, J., Cockburn, A. (2001). Agile software development: The business of innovation. IEEE Computer. Vol. 34(9). pp. 120-127.

Hodgetts, P. (2005). Experiences integrating sophisticated user experience design practices into agile processes. Proceedings of the Agile Development Conference. IEE Computer Society. pp. 235-242.

ISO 13407. (1999). Human-centered design processes for interactive system. International Organization for Standardization.

ISO 9241-11. (1998). Ergonomic, Part 11: Guidance for usability. International Organization for Standardization.

ISO 9241-210. (2009). Ergonomics of human system integration – Part 210: Human centered design for interactive systems. International Organization for Standardization.

Kane, D. (2003). Finding a place for discount usability engineering in agile development: Throwing down the gauntlet. Proceedings of the Conference on Agile Development. IEEE Computer Society.

Kapor, M. (1990). Software design manifesto.

Korstjens, I., Moser, A. (2018). Series: Practical guidance to qualitative research. Part 4: Trustworthiness and publishing. European journal of general practice. Vol. 24(1). pp. 120-124.

Kuusinen, K., Väänänen-Vainio-Mattila, K. (2012). How to make agile UX work more efficient: Management and sales perspectives. NordiHCI. pp. 139-148.

Kyng, M. (1995). Making representations work. Communication of the ACM. Vol. 38(9). pp. 46-55.

Lárusdóttir, M., Cajander, Å., Gulliksen, J. (2013). Informal feedback rather than performance measurements – user-centred evaluation in Scrum projects. Behaviour & Information Technology. Vol. 33(11). pp. 1118-1135.

Lárusdóttir, M., Gulliksen, J., Cajander, Å. (2017). A license to kill – Improving UCSD in agile development. The Journal of Systems and Software. Vol. 123. pp.214-222.

Lievesley, M. A., Yee, J. (2006). The role of the interaction designer in an agile software development process. CHI ’06 Extended Abstract on Human Factors in Computing Systems. pp. 1025-1030.

Lievesley, M. A., Yee, J. (2007). Surrogate users: A pragmatic approach to defining user needs. CHI ’07 Extended Abstract on Human Factors in Computing Systems. pp. 1789-1794.

Lincoln, Y. S., Guba, E. G. (1985). Naturalistic inquiry. California: Sage Publications.

Miller, L. (2005). Case study of customer input for a successful product. 2015 Agile Conference. Dencer. CO. USA. IEEE Computer Society. pp. 225-234.

Nielsen, J. (1993). Usability engineering. Cambridge, MA: AP Professional.

Opelt, A., Gloger, B., & Pfarl, W. (2013). Agility: What is that? Agile contracts: Creating and managing successful projects with scrum. pp. 1-32.

Peixoto, C. S. A., Da Silva, A. E. A. (2009). A conceptual knowledge base representation for agile design of human-computer interface. Proceedings of the 3rd International Conference on Intelligent Information Technology Application. pp. 156-160.

Salah, D., Paige, R. F., Cairns, P. (2014). A systematic literature review for agile development processes and user centred design integration. Proceedings of the 18th International Conference on evaluation and assessment in software engineering.

Schwaber & Sutherland (2017). The Scrum Guide. The definitive guide to Scrum: The rules of the game. WWW-julkaisu. Saatavilla: https://www.scrum.org/resources/scrum-

guide (Viitattu 03.06.2019).

Shenton, A. K. (2004). Strategies for ensuring trustworthiness in qualitative research projects. Education for Information. Vol. 24. pp. 63-74.

Singh, M. (2008). U-SCRUM: An agile methodology for promoting usability. Agile Conference. pp. 555-560.

Sy, D. (2007). Adapting usability investigations for agile user-centered design. Journal of Usability studies. Vol. 2(3). pp.112-132.

Tzanidou, K., Ferreira, J. (2010). Design and development in the agile room: Trialing scrum at a digital agency. In Sillitti, A., Martin, A., Wang X, Whitworth, E. (ed.) Agile Processes in Software Engineering and Extreme Programming. Lecture Notes in Business Information Processing. Vol. 48. Springer Berlin Heidelberg. pp. 372-378.

VersionOne. (2019). 13th annual state of agile report. WWW-julkaisu. Saatavilla: https://www.stateofagile.com/#ufh-i-521251909-13th-annual-state-of-agile-report/473508 (Viitattu 18.06.2019).